AFRL-IF-RS-TR-2005-35
Final Technical Report
February 2005

INFERENCING IN SUPPORT OF ACTIVE TEMPLATES

University of Maryland

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. J859

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** February 2005	**3. REPORT TYPE AND DATES COVERED** **FINAL Jan 00 – Apr 04**

4. TITLE AND SUBTITLE

INFERENCING IN SUPPORT OF ACTIVE TEMPLATES

5. FUNDING NUMBERS

G - F30602-00-2-0505
PE - 63760E
PR - ATEM
TA - P0
WU - 02

6. AUTHOR(S)

Dana Nau

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Maryland
Department of Computer Science
College Park MD 20742

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington VA 2203-1714

AFRL/IFTB
525 Brooks Road
Rome NY 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2005-35

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Richard A. Hyle/IFSB/(315) 330-4857 Richard.Hyle@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*

The primary accomplishments of this project include the following:

(1) HICAP is a general purpose planning architecture to assist military commanders with planning NEOs (Noncombatant Evacuation Operations). HICAP integrates a hierarchical task editor with a planning tool. Military planners select a task to decompose in the task editor and then use the planning tool to interactively refine it into an operational plan.

(2) SHOP and SHOP2 are some simple, practical planning tools based on HTN planning. Their practical utility is shown by the mergence of an active set of users, which include government laboratories, industrial R&D projects, and academic settings. SHOP2 received one of the top four awards in the 2002 International Planning Competition.

(3) The Air Force Research Laboratory's Causal Analysis Tool (CAT) is a system for creating and analyzing causal models similar to Bayesian networks. We have enhanced CAT by developing an approach to quickly generate plans that have high probabilities of success.

14. SUBJECT TERMS

Planning, Task Support, Coalition, Collaboration, AI

15. NUMBER OF PAGES 50

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

Chapter 1

Summary

This is the final report for AFRL Contract F30602-00-2-0505, *Inferencing in Support of Active Templates*. The primary accomplishments of this project are as follows:

HICAP. HICAP is a general purpose planning architecture that we have developed and applied to assist military commanders with planning *NEOs* (Noncombatant Evacuation Operations). HICAP integrates a hierarchical task editor, HTE, with SiN (SHOP in NaCoDAE), a planning tool that tightly integrates conversational case-based planning (through HICAP's NaCoDAE/HTN subsystem) with HTN decomposition (using the SHOP planner described below). In this application, HTE maintains an agenda of tactical planning tasks that, according to military doctrine, must be addressed in a NEO plan. It also supports several bookkeeping tasks, which are crucial for large-scale planning tasks that differ greatly among different NEO operations. Military planners select a task to decompose from HTE and then use SiN to interactively refine it into an operational plan by selecting and applying cases, which represent task decompositions from previous NEO operations. Thus, HICAP helps military commanders by using previous experience to formulate operational plans that are in accordance with NEO doctrine.

SHOP and SHOP2. SHOP and SHOP2 are HTN planning systems that we designed with two goals in mind: to investigate some research issues in automated planning, and to provide some simple, practical planning tools that could be used to support AcT and also be used elsewhere. We have made them available as open-source software, and they have been downloaded thousands of times. Their practical utility is shown by the emergence of an active set of users, which include government laboratories, industrial R&D projects, and academic settings. As an example of their research impact, SHOP2 received one of the top four awards in the 2002 International Planning Competition.

CAT. The Air Force Research Laboratory's Causal Analysis Tool (CAT) is a system for creating and analyzing causal models similar to Bayesian networks. To use CAT as a tool for planning military operations, users go through an iterative process in which they use CAT to create alternative plans and then use CAT to analyze these plans. One of the biggest difficulties is that there is an exponentially large number of possible plans, making it impossible for the user to create and analyze every possible plan.

To solve this problem, we have developed an approach to quickly compute upper and lower bounds on the probabilities of success associated with a partial plan, and use these probabilities to recommend which actions the user should include in the plan in order to get a complete plan. This provides an exponential reduction in the amount of time needed to find a complete plan. In our experiments, our approach generated recommendations that resulted in plans that have the highest probability of success in just a few minutes.

Chapter 2

Introduction

The introduction is divided into three parts: one for HICAP, one for SHOP and SHOP2, and one for CAT.

2.1 HICAP

Generative planners traditionally require a complete description of the planning domain. However, in practical planning applications, developing a complete description is not always feasible.

One example is the task of planning a Noncombatant Evacuation Operation (NEO). These are military evacuation operations that require performing hundreds of subtasks and whose primary goal is to minimize loss of life. Formulating a NEO plan is a complex task because it involves considering a wide range of factors (e.g., military resources, political issues, meteorological predictions) and uncertainties (e.g., hostility levels and locations). Flawed NEO plans could yield dire consequences. For example, Siegel [62] reported a mistake during Eastern Exit. Doctrine states that evacuees must be inspected prior to embarkation in military transports. However, this task was never assigned to any of the ground teams, and one of the evacuees produced his weapon during a helicopter evacuation flight. Although it was immediately confiscated, this oversight may have resulted in tragedy and illustrates the difficulties with planning NEOs manually.

In general, there will be an incomplete domain description, in the form of standard requirements and operating procedures. However, these cannot be used to derive detailed plans, which often require knowledge about previous experiences.

Formulating a NEO can be quite complex. Typically there will be hundreds of tasks to be carried out. These tasks will depend on a wide range of factors: sources of danger, available resources, geography, weather predictions, political issues, and so forth. Complete information about the current state will never be available; the planning must include dynamic information gathering, and plans must be formulated with an incomplete world state.

For such a problem, the planning must be done by a human expert or under the supervision of a human expert. It is unrealistic to expect that a planning system could produce good plans by itself, and flawed evacuation plans could yield dire consequences.

HICAP is a general purpose planning architecture that we have developed and applied to assist military commanders with planning *NEOs* (Noncombatant Evacuation Operations). HICAP integrates a hierarchical task editor with a planning tool that tightly integrates conversational case-based planning with HTN decomposition. HICAP maintains an agenda of tactical planning tasks that, according to military doctrine, must be addressed in a NEO plan. It also supports several bookkeeping tasks, which are crucial for large-scale planning tasks that differ greatly among different NEO operations. Military planners select a task to decompose and interactively refine it into an operational plan by selecting and applying cases, which represent task decompositions from previous NEO operations. Thus, HICAP helps military commanders by using previous experience to formulate operational plans that are in accordance with NEO doctrine.

2.2 SHOP and SHOP2.

SHOP and SHOP2 are HTN planning systems that we designed with two goals in mind: to investigate some research issues in automated planning, and to provide some simple, practical planning tools that could be used to support AcT and also be used elsewhere. They have been successful in both respects.

SHOP and SHOP2 are available as open-source software, and have been downloaded thousands of times. Their practical utility is shown by the emergence of an active set of users, which include government laboratories, industrial R&D projects, and academic settings. As an example of their research impact, SHOP2 received one of the top four awards in the 2002 International Planning Competition [25].

One reason for the success of SHOP and SHOP2 is their use of Hierarchical Task Networks (HTNs). HTN planning is done by applying *HTN methods*, which basically are forms that describe how to decompose tasks into subtasks. HTN methods can be used to describe the "standard operating procedures" that one would normally use to perform tasks in some domain; thus they often correspond well to the way that users think about problems.

Another reason for the success of SHOP and SHOP2 is their use of a search-control strategy called *ordered task decomposition*, which reduces the complexity of reasoning by eliminating a great deal of uncertainty about the world. Ordered task decomposition makes it easy to incorporate a great deal of expressive power into the planning system: for example, SHOP and SHOP2 can do complex inferential reasoning, mixed symbolic/numeric computations, and call user-supplied subroutines.

2.3 CAT

In planning a course of action (i.e., a plan to achieve a desired objective or objectives), quick and accurate decision making is a very important task and it is very hard. A major source of difficulty is how to deal with uncertainty. This uncertainty has many sources, but perhaps the biggest one is the uncertain relationship between causes and effects. For example:

- *At a tactical level, sorties are flown against a series of bridges to prevent the enemy ground forces from crossing the river. The sorties are intended to prevent the crossing. What is the probability that they will?*
- *At a strategic level, the destruction of the Taliban Army was intended ultimately to reduce world-wide terrorism. Did it?*

Such uncertainties are compounded by the size and complexity of most military plans—for example, a causal model of Operation Deny Freedom, built by the actual planners, contains over 300 uncertain events interrelated by cause and effect. Moreover, there are often significant delays between cause and effect, and effects may persist for only limited amounts of time: a destroyed bridge can be rebuilt or bypassed. This makes it exceedingly difficult to forecast the possible effects of a military operation.

We have developed a way to help analyze this uncertainty in order to generate effective plans. The basis for our approach is the Air Force Research Laboratory's (AFRL's) Causal Analysis Tool (CAT), which is a tool for representing and analyzing causal networks similar to Bayesian networks. From this representation, CAT can compute the probability that any given plan (i.e., any chosen combination of actionable items) will achieve the desired objectives.

A major technical difficulty is how to overcome combinatorial blowup during the planning process. If there are n different actionable items, then there are potentially 2^n different plans, making it infeasible for the user to ask CAT to analyze each one. Our approach exploits the conditional-independence relationships within a causal network in order to overcome this combinatorial blowup. In doing so, it quickly computes upper and lower bounds on the probabilities of success associated with a partial plan, and uses these bounds to recommend which actions the user should include in the plan in order to get a complete course of action. This provides an exponential reduction in amount of time needed to find a complete plan. In our experimental evaluation, our approach generated recommendations that resulted in plans that have the highest probability of success in just a few minutes, demonstrating its effectiveness.

Chapter 3

Methods, Assumptions, and Procedures

This section is divided into three parts: one for HICAP, one for SHOP and SHOP2, and one for CAT.

3.1 HICAP

Generative planners traditionally require a complete description of the planning domain. However, in practical planning applications, developing a complete description is not always feasible.

One example is the task of planning a Noncombatant Evacuation Operation (NEO). These are military evacuation operations that require performing hundreds of subtasks and whose primary goal is to minimize loss of life. Formulating a NEO plan is a complex task because it involves considering a wide range of factors (e.g., military resources, political issues, meteorological predictions) and uncertainties (e.g., hostility levels and locations). Flawed NEO plans could yield dire consequences. For example, Siegel [62] reported a mistake during Eastern Exit. Doctrine states that evacuees must be inspected prior to embarkation in military transports. However, this task was never assigned to any of the ground teams, and one of the evacuees produced his weapon during a helicopter evacuation flight. Although it was immediately confiscated, this oversight may have resulted in tragedy and illustrates the difficulties with planning NEOs manually.

In general, there will be an incomplete domain description, in the form of standard requirements and operating procedures. However, these cannot be used to derive detailed plans, which often require knowledge about previous experiences.

Formulating a NEO can be quite complex. Typically there will be hundreds of tasks to be carried out. These tasks will depend on a wide range of factors: sources of danger, available resources, geography, weather predictions, political issues, and so forth. Complete information about the current state will never be available; the planning must include dynamic information gathering, and plans must be formulated with an incomplete world state.

For such a problem, the planning must be done by a human expert or under the supervision of a human expert. It is unrealistic to expect that a planning system could produce good plans by itself, and flawed evacuation plans could yield dire consequences.

This section describes a plan formulation tool, HICAP (Hierarchical Interactive Case-Based Architecture for Planning), that was designed to assist human experts in planning emergency evacuations. Since the plans are strongly hierarchical in nature, HICAP represents plans using Hierarchical Task Networks (HTNs).

As shown in Figure 3.1, HICAP integrates a task decomposition editor, HTE, with a mixed-initiative planning system, SiN. HTE allows users to edit tasks, and SiN allows users to interactively refine HTN plans. Their integration in HICAP ensures that operational plans are framed within the standard requirements and operating procedures or within the changes made by human planners through interactive task editing and and interactions with HICAP's case-based and generative planning modules.

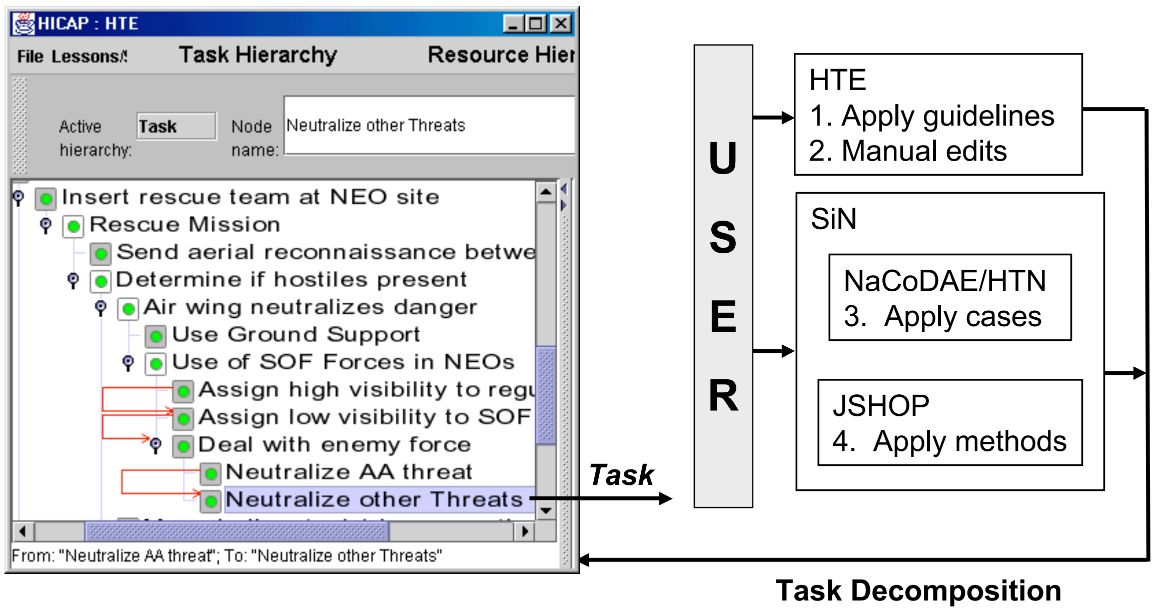

HICAP : HTE
File Lessons/
Task Hierarchy
Resource Hier
Active hierarchy: Task
Node name: Neutralize other Threats
Insert rescue team at NEO site
Rescue Mission
Send aerial reconnaissance betwe
Determine if hostiles present
Air wing neutralizes danger
Use Ground Support
Use of SOF Forces in NEOs
Assign high visibility to regu
Assign low visibility to SOF
Deal with enemy force
Neutralize AA threat
Neutralize other Threats
From: "Neutralize AA threat"; To: "Neutralize other Threats"
Task
USER
HTE
1. Apply guidelines
2. Manual edits
SiN
NaCoDAE/HTN
3. Apply cases
JSHOP
4. Apply methods
Task Decomposition

Figure 3.1: The HICAP plan-authoring system.

The following sections describe the application domain, HICAP's knowledge representation, and its architecture.

3.1.1 Evacuation Operations

NEOs are conducted to assist the USA Department of State (DoS) in evacuating noncombatants, nonessential military personnel, selected host-nation citizens, and third country nationals whose lives are in danger from locations in a host foreign nation to an appropriate safe haven. They usually involve a swift insertion of a force, temporary occupation of an objective (e.g., a USA Embassy), and a planned withdrawal after mission completion. NEOs are usually planned and operated by a Joint Task Force (JTF) and conducted under a USA Ambassador's authority. Force sizes can range into the hundreds with all branches of armed services involved, while the evacuees can number into the thousands. At least ten NEOs were conducted within the past decade (e.g., Eastern Exit (1991, Mogadishu, 300 evacuees), Assured Response (1997, Monrovia, 2400)). Unclassified publications exist that describe NEO doctrine (e.g., DoD, 1994), case studies (e.g., Siegel [62, 63]), and more general analyses [67, 41].[1]

The decision making process for a NEO is made at three increasingly-specific levels: strategic, tactical, and operational. The strategic level involves global and political considerations such as whether to perform the NEO. The tactical level involves considerations such as determining the size and composition of the force executing the NEO. The operational level is the concrete level, which assigns specific resources to specific tasks.

JTF Commanders (CJTF) plan NEOs in the context of doctrine [19], which establishes a framework for designing strategic and tactical plans; operational considerations are only partly addressed. Doctrine describes general aspects that must be considered when planning a military operation, including the chain of command and a series of tasks to perform. However, doctrine is limited; it is idealized and cannot account for characteristics of specific NEOs. Thus, the CJTF must always adapt doctrine to the specific needs of a NEO, and does so in two ways. First, he must dynamically modify doctrinal guidance by eliminating

[1]See http://www.aic.nrl.navy.mil/~aha/neos for more information on NEOs.

irrelevant planning tasks and adding others, depending on the operation's needs, resource availabilities, and the planners' past experiences. For example, although NEO doctrine states that a forward command element (FCE) must be inserted into the evacuation area prior to the primary evacuation elements, temporal constraints sometimes prevent this insertion (e.g., [62]). Second, he must employ experiences from previous NEOs, which complement doctrine by suggesting operational refinements that are suitable for the current environment. For example, he could draw upon his own experience or from others to identify whether it is appropriate to concentrate the evacuees in the embassy or to plan on multiple evacuation sites. In summary, military planners use guidance from both doctrine and their operational experiences (i.e., from past operations and exercises) to plan NEOs.

The following sections describe how HICAPis intended to assist human planners by interactively developing evacuation plans.

3.1.2 Knowledge Representation

HICAP uses a variant of hierarchical task networks (HTNS) [27, Chapter 11], which are descriptions of possible ways to accomplish tasks by breaking them down into subtasks to be accomplished. It also uses *cases*, portions of plans that were formulated during previous planning episodes [27, Section 24.2]. Both of these are described below.

HTNs

In HICAP, an HTN is a set of tasks and their ordering relations, denoted as $N = (\{t_1, \ldots, t_m\}, \prec)$, where $m \geq 0$ and $\prec$ is a binary relation expressing temporal constraints between tasks. Decomposable tasks are called *compound*, while non-decomposable tasks are called *primitive*.

A domain description consists of methods and operators for generating plans. A method is an expression of the form $M = (h, P, ST)$, where h (the method's head) is a compound task, P is a set of preconditions, and ST is the set of M's subtasks. M is applicable to a task t, relative to a state S (a set of ground atoms), iff $matches(h, t, S)$ holds (i.e., h and t have the same predicate and arity, and a consistent set of bindings B exists that maps variables to values such that all terms in h match their corresponding ground terms in t) and the preconditions P are satisfied in S.

An operator is an expression of the form $O = (h, aL, dL)$, where h (the operator's head) is a primitive task, and aL and dL are the add- and delete-lists. These specify that when the operator is executed, every element in the add-list is to be added to S and every element in the delete-list is to be removed from S. An operator O is applicable to a task t, relative to a state S, iff $matches(h, t, S)$.

A planning problem is a triple (T, S, D), where T is a set of tasks, S is a state, and D is a domain description. A plan is the collection of primitive tasks obtained by decomposing all compound tasks in a planning problem (T, S, D).

Cases

In many domains it is impossible to assume that a complete domain description of the world is known. A partial domain description may exist, in the form of standard requirements and operating procedures—and these can be encoded into methods and operators.

For those parts of the domain for which no domain description is available, reasoning is done through *cases*. In HICAP case is a task decomposition that was created by the user while solving a previous planning problem. A case looks similar to an instance of a method, but usually it is not an instance of any method in the domain description.

Syntactically, a case is denoted by $C = (h, P, ST, Q)$, where h, P, and ST are defined as for methods and Q is a set of $(question, answer)$ pairs. Q defines preferences for matching a case to the current state.

Preferences are useful for ranking cases in the context of incomplete world states and/or domain theories because they focus users on providing relevant additional state information.

3.1.3 Hierarchical Task Editor

Because NEOs can be complex, it is difficult for the CJTF and his staff to keep track of the completion status for each task to be performed and each element of the JTF. The *Hierarchical Task Editor* (HTE) was conceived to facilitate the NEO planning process. Given a domain-specific knowledge base for tactical planning, HTE can be used to:

1. browse and edit the knowledge base's components,
2. select tasks for further decomposition, and
3. investigate the status of tasks.

HTE serves HICAP as a bookkeeping tool; it maintains the task agenda and helps planners to formulate plans for decomposable tasks.

HTE's knowledge base consists of a HTN, a command hierarchy, and an assignment of tasks to commands. For our application to NEO plan formulation, we encoded an HTN to capture critical planning knowledge corresponding to NEO doctrine [19]. This required a substantial manual knowledge acquisition effort; our HTN consists of more than 200 tasks and their ordering relations. Next, we elicited the JTF command hierarchy that is commonly used in NEO operations. The elements in the JTF are represented in a tree where each node denotes a military commander and its children denote the commander's subordinates. Finally, we elicited relations between tasks and the elements in the JTF responsible for them. This is represented by an *assignment* function from the elements of the JTF to the tasks because the mapping of tasks to command elements is many-to-one.

In addition to providing users with a visual description of the standard requirements and procedures, HTE can be used to edit the HTN, its ordering relations, the command hierarchy, and the mapping between tasks and command assignments. Thus, military commanders can use HTE to tailor its knowledge base according to the particular circumstances of the current operation. Furthermore, they can modify the command hierarchy as needed to represent the resources available for the current planning scenario. Finally, they can reassign tasks and/or command elements.

Figure 3.2 displays the top level tasks that, according to doctrine, must be performed during a NEO and the elements in the JTF responsible for them. ISB denotes the *intermediate stage base*, the location where the JTF is based prior to the evacuation. SH denotes the *safe haven* where the evacuees will be transported. Arrows between tasks denote their execution ordering.

3.1.4 SiN

HICAP incorporates a mixed-initiave planner, SiN (SHOP integrated with NaCoDAE). SiN is a synthesis of JSHOP, a generative planner, with NaCoDAE, a conversational case retriever [14]. SiN is a provably correct algorithm that does not require a complete domain description nor complete information about initial or intermediate world-states.

Users can interact with HTE by selecting a task T to be decomposed. This invokes SiN to start decomposing the task under supervision of the user. This decomposition can be recursive; subtasks of N can themselves be decomposed further. Eventually, non-decomposable tasks corresponding to operational actions will be reached. Task decompositions are immediately displayed by HTE.

The SiN planning algorithm integrates the task decomposition algorithms of two planning systems: the JSHOP generative planner and the NaCoDAE case-based planner. A single (current) state S is maintained in SiN that is accessible to and updateable by both JSHOP and NaCoDAE. Answers given by the user during

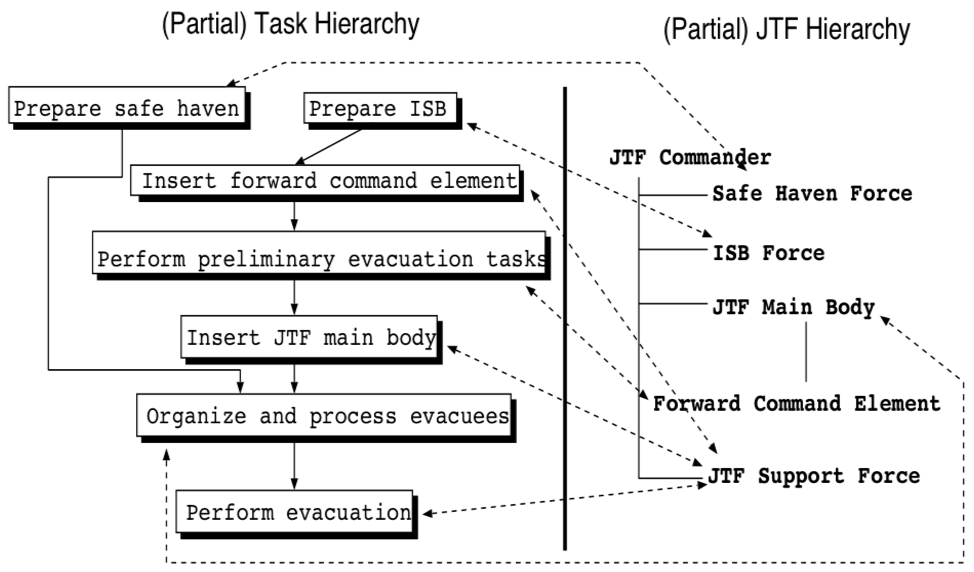


Figure 3.2: Top-level tasks.

an interaction with NaCoDAE are added to S (i.e., each question has a translation into a ground atom). Changes to the state that occur by applying JSHOP's operators are also reflected in S.

JSHOP is a Java implementation of SHOP [58], which is described in Section 3.2.[2]

NaCoDAE is a mixed-initiative case retriever. Users interact with NaCoDAE in *conversations*, which begin when the user selects a task t. NaCoDAE responds by displaying the top-ranked cases whose pre-conditions are satisfied and whose heads match t. Cases are ranked according to their similarity to the current state S, which is the state that exists at that time during the conversation. Similarity is computed for each case C by comparing the contents of S with Q, C's (q, a) preference pairs. (That is, each pair is represented as a monadic atom in S, and similarity for a given (q, a) preference pair becomes a membership test in S). NaCoDAE also displays questions, whose answers are not known in S, ranked according to their frequency among the top-ranked cases. The user can select and answer (with a) any displayed question q, which inserts (q, a) into S. This state change subsequently modifies the case and question rankings. A conversation ends when the user selects a case C, at which time the task t is decomposed into ST (i.e., C's subtasks).

SiN receives as input a set of tasks T, a state S, and a knowledge base $I \cup B$ consisting of an incomplete domain description I and a collection of cases B. The output is a solution plan π consisting of a sequence of operators in I. Both JSHOP and NaCoDAE assist SiN with refining T into a plan. As does JSHOP, SiN maintains the set of tasks in T' that have not been decomposed and the partial solution plan π. At any point of time, either JSHOP or NaCoDAE is in control and is focusing on a compound task $t \in T'$ to decompose. SiN proceeds as follows:

- Rule 1: If JSHOP is in control and can decompose t, it does so and retains control. If JSHOP cannot decompose t, but NaCoDAE has cases for decomposing t, then JSHOP will cede control to NaCoDAE.
- Rule 2: If NaCoDAE is in control, it has cases for decomposing t whose pre-conditions are satisfied. If the user applies one of them to decompose t, then NaCoDAE retains control. If NaCoDAE has no cases to decompose t or if the user decides not to apply any applicable case, then if t is JSHOP-decomposable, NaCoDAE will cede control to JSHOP.

[2]JSHOP is available for downloading at http://www.cs.umd.edu/projects/shop.

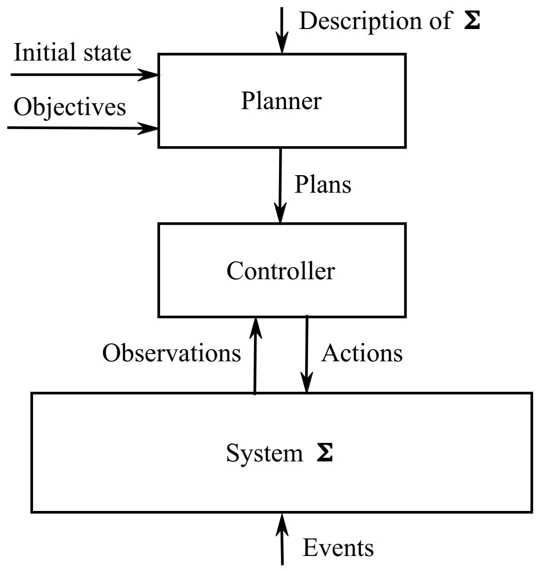


Figure 3.3: A simple conceptual model for planning. Σ is the system for which the planner is formulating its plans.

If neither of these rules applies, then SiN backtracks, if possible. If backtracking is impossible (e.g., because t is a task in T), this planning process is interrupted and a failure is returned.

By continuing in this way, and assuming that the process is not interrupted with a failure, SiN will eventually yield a plan π.

3.2 SHOP and SHOP2

The SHOP and SHOP2 planning systems were designed with two goals in mind: to investigate some research issues in automated planning, and to provide some simple, practical planning tools. They have been successful in both respects.

SHOP and SHOP2 are available as open-source software, and have been downloaded thousands of times. Their practical utility is shown by the emergence of an active set of users, which include government laboratories, industrial R&D projects, and academic settings. As an example of their research impact, SHOP2 received one of the top four awards in the 2002 International Planning Competition [25].

One reason for the success of SHOP and SHOP2 is their use of Hierarchical Task Networks (HTNs). HTN planning is done by applying *HTN methods*, which basically are forms that describe how to decompose tasks into subtasks. HTN methods can be used to describe the “standard operating procedures” that one would normally use to perform tasks in some domain; thus they often correspond well to the way that users think about problems.

Another reason for the success of SHOP and SHOP2 is their use of a search-control strategy called *ordered task decomposition*, which reduces the complexity of reasoning by eliminating a great deal of uncertainty about the world. Ordered task decomposition makes it easy to incorporate a great deal of expressive power into the planning system: for example, SHOP and SHOP2 can do complex inferential reasoning, mixed symbolic/numeric computations, and call user-supplied subroutines.

3.2.1 Background

Automated Planning. In general, the purpose of an automated planning system is to generate a plan or policy that a plan executor can execute in order to achieve some set of goals or objectives. Most planning research has focused on *offline* planning (see Figure 3.3), in which the entire plan is formulated before the plan executor begins executing it. Thus at planning time, no direct information is available to the planner about a plan's success or failure—instead, the planner must reason about whether the plan will (or is likely to) succeed or fail. Automated planning systems can be classified roughly into three types:

- *Domain-specific planning systems*, where the planning domain is known beforehand and the system is designed specifically to reason about plans in that domain. Several of the most successful planning systems are of this type (e.g., [66]).
- *Domain-independent planning systems*, which are designed to work in any domain within some large class of domains (e.g., the well-known *classical* planning domains [27]), provided that the input includes definitions of the basic actions in the domain. Domain-independent systems have been developed that work quite well in abstract domains—but getting them to work well in domains of practical importance has been an elusive goal.
- *Domain-configurable planning systems*. Here, the planning engine is domain-independent, and the domain description includes both the basic actions (like in domain-independent planning) and also some information about how those actions should or may be combined in order to solve planning problems.

Much more work has been done on automated planning than we can describe here, and we refer the reader to [27] for details.

HTN Planning. For domain-specific and domain-configurable planning, one of the best-known approaches is *HTN planning*, in which the planning system formulates a plan by decomposing *tasks* (symbolic representations of activities to be performed) into smaller and smaller subtasks until *primitive* tasks are reached that can be performed directly. The basic idea was developed in the mid-70s [61, 68], and the formal underpinnings were developed in the mid-90s [22].

HTN-planning research has been much more application-oriented than most other AI-planning research. Most of the domain-configurable systems (e.g., O-Plan [69], SIPE-2 [75], SHOP [58], and SHOP2 [57]) have been used in application development, and domain-specific HTN planning systems have been built for several application domains (e.g., [66]).

An HTN planning problem consists of the following: the *initial state* (a symbolic representation of the state of the world at the time that the plan executor will begin executing its plan), the *initial task network* (a set of tasks to be performed, along with some constraints that must be satisfied), and a *domain description* that contains the following:

- A set of *planning operators* that describe various kinds of actions that the plan executor can perform. Each operator may have a set of preconditions that must be true in the state in which the operator is to be executed, and a set of effects that will occur when the operator is executed. Each of the possible actions is an operator instance, produced by assigning values to an operator's parameters.
- A set of *methods* that describe various possible ways of decomposing tasks into subtasks. These are the "standard operating procedures" that one would normally use to perform tasks in the domain. Each method may have a set of constraints that must be satisfied in order to be applicable.
- Optionally, various other information such as definitions of auxiliary functions and definitions of axioms for inferring conditions that are not mentioned explicitly in states of the world.

Planning is done as follows. For each nonprimitive task, the planner chooses an applicable method and

```
*method* travel-by-foot
  precond:  $distance(x,y) \leq 2$
  task:     travel($a,x,y$)
  subtasks: walk($a,x,y$)

*method* travel-by-taxi
  task:     travel($a,x,y$)
  precond:  $cash(a) \geq 1.5 + 0.5 \times distance(x,y)$
  subtasks: call-taxi($a,x$) $\longrightarrow$ ride($a,x,y$) $\longrightarrow$ pay-driver($a,x,y$)

*operator* walk
  precond: $location(a) = x$
  effects: $location(a) \leftarrow y$

*operator* call-taxi($a,x$)
  effects: $location(taxi) \leftarrow x$

*operator* ride-taxi($a,x$)
  precond: $location(taxi) = x$, $location(a) = x$
  effects: $location(taxi) \leftarrow y$, $location(a) \leftarrow y$

*operator* pay-driver($a,x,y$)
  precond: $cash(a) \geq 1.5 + 0.5 \times distance(x,y)$
  effects: $cash(a) \leftarrow cash(a) - 1.5 + 0.5 \times distance(x,y)$
```

Figure 3.4: Pseudocode representation of an extremely simple travel-planning domain. Left-arrows denote assignments of values to state-variables; right-arrows are ordering constraints.

instantiates it to decompose the task into subtasks. For each primitive task, the planner chooses an applicable operator and instantiates it to produce an action. If all of the constraints are satisfied, then the planner has found a solution plan; otherwise the planning system will need to backtrack and try other methods or other instantiations.

Example. Figure 3.4 gives a pseudocode representation of an extremely simple planning domain in which there are two ways to travel from one location to another: by foot and by taxi. These are represented by two methods: travel-by-foot and travel-by-taxi. The travel-by-foot method has one constraint: a precondition saying that the distance from the starting point to the destination must be less than or equal to 2 miles. If the method is applicable, it decomposes the task into a single subtask: walk to the park. The travel-by-taxi method has one constraint, which is also a precondition: the traveler must have enough cash to pay the taxi driver. If the method is applicable, it decomposes the task into three subtasks: call a taxi, ride to the park, and pay the driver. All of the subtasks are primitive, i.e., the traveler is expected to know how to accomplish them directly.

Now, suppose that in the initial state, I am at home, I have \$20, and I want to travel to a park that is 8 miles away. To plan how to travel to the park (see Figure 3.5), first I try to use the travel-by-foot method, but this method is not applicable because the park is more than 2 miles away. Next, I try to use the travel-by-taxi method. Its precondition is satisfied, so the method produces a sequence of three subtasks, with a constraint saying they are to be performed in the following order: (1) call a taxi to my home, (2) ride in it to the park, and (3) pay the driver \$5.50. The subtasks all are primitive, i.e., each of them corresponds to an action. The first action has no preconditions, so it is applicable and produces a state s_1 that is identical to the initial state except that $location(taxi) = home$. This state satisfies the preconditions of the second action. The second action produces a state in which the precondition of the third action is satisfied, so I have a solution plan.

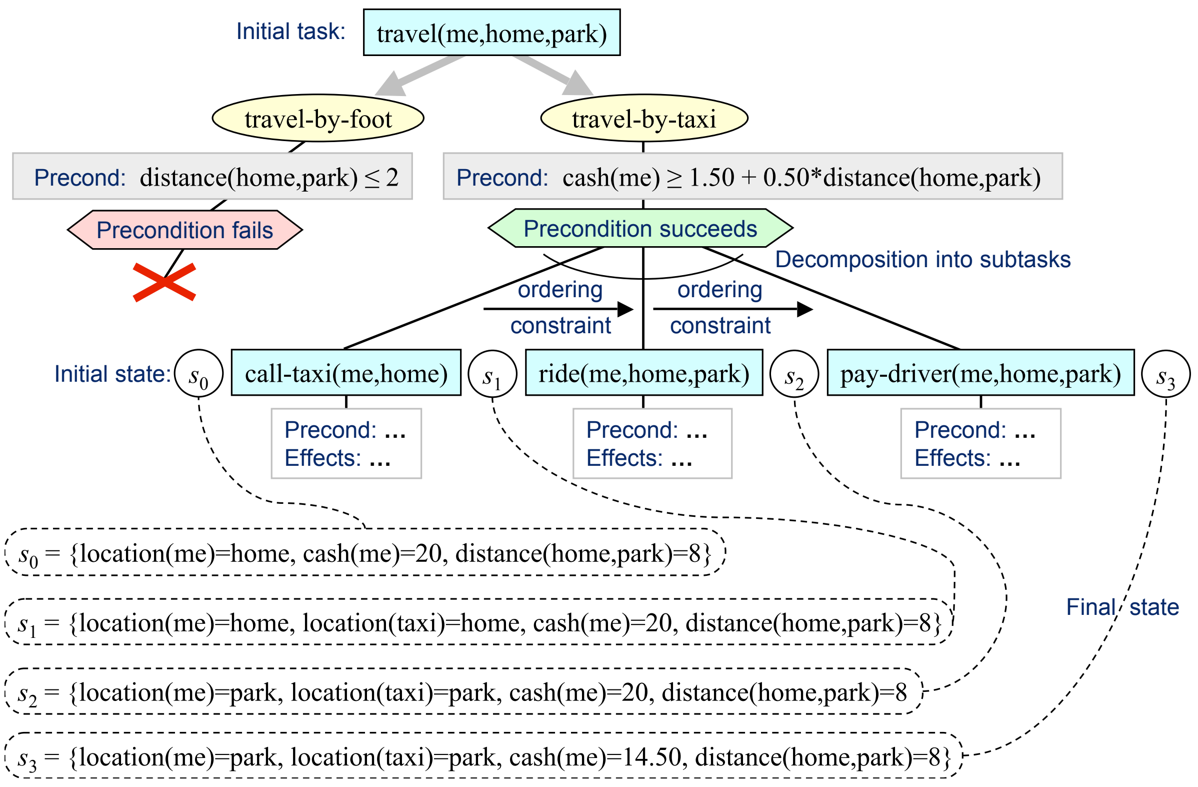

Initial task:
travel(me,home,park)
travel-by-foot
travel-by-taxi
Precond: distance(home,park) ≤ 2
Precond: cash(me) ≥ 1.50 + 0.50*distance(home,park)
Precondition fails
Precondition succeeds
Decomposition into subtasks
ordering constraint
ordering constraint
Initial state:
s_0
call-taxi(me,home)
s_1
ride(me,home,park)
s_2
pay-driver(me,home,park)
s_3
Precond: ...
Effects: ...
Precond: ...
Effects: ...
Precond: ...
Effects: ...
s_0 = {location(me)=home, cash(me)=20, distance(home,park)=8}
s_1 = {location(me)=home, location(taxi)=home, cash(me)=20, distance(home,park)=8}
s_2 = {location(me)=park, location(taxi)=park, cash(me)=20, distance(home,park)=8
s_3 = {location(me)=park, location(taxi)=park, cash(me)=14.50, distance(home,park)=8}
Final state

Figure 3.5: Solving a planning problem in the travel-planning domain.

After execution of this plan, the final state will be as shown in the figure.

3.2.2 How SHOP and SHOP2 Work

HTN planning is basically a trial-and-error search: the planner may have to try many different possibilities before finding a plan that works. In any trial-and-error search, one of the most important questions is what kind of search-control strategy to use.

SHOP and SHOP2 use a search-control strategy called *ordered task decomposition*: they choose to decompose tasks into subtasks in the same order as the order in which the tasks are supposed to be accomplished. As a consequence, SHOP and SHOP2 generate the steps of each plan in the same order that the plan executor will execute those steps (see Figure 3.6), so they know the current state at each step of the planning process. This reduces the complexity of reasoning by eliminating a great deal of uncertainty about the world, thereby making it easy to incorporate substantial expressive power into the planning system, such as the auxiliary functions and axioms mentioned earlier.

The primary difference between SHOP and SHOP2 is that SHOP requires a strict linear ordering on subtasks and does not allow them to be interleaved. In contrast, SHOP2 does not impose these requirements. For example, in Figure 3.6, the subtasks of task t_3 and task t_5 are interleaved; this can occur in SHOP2 but not in SHOP. As a result, some planning domains that would be rather cumbersome to describe in SHOP can be described more easily in SHOP2.

In April 2002, the SHOP2 planning system achieved high visibility because of its performance in the 2002 International Planning Competition [25], where it received one of the top four awards.[3] SHOP2 was

[3]There were two awards for “distinguished performance” and two for “distinguished performance of the first order;” SHOP2 received one of the former. For more information about the competition, see ⟨http://planning.cis.strath.ac.uk/competition⟩.

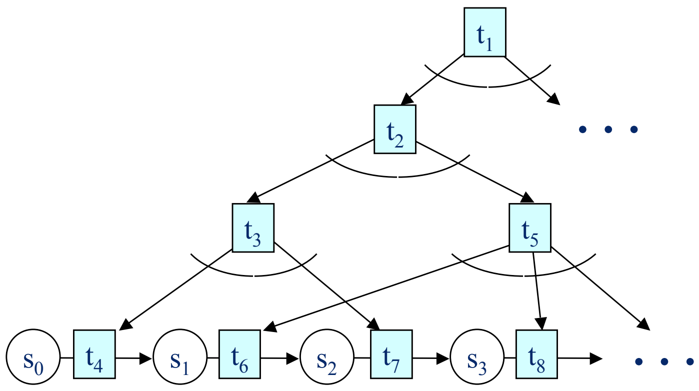


Figure 3.6: An example of ordered task decomposition. The subscripts show the order in which the tasks are decomposed. In this example, tasks t_4, t_6, t_7, t_8 are primitive, i.e., they correspond to planning operators.

one of the three fastest planners in the competition: it was able to solve planning problems many times faster and many times more complicated than those solved by most of the other systems. In addition, SHOP2 solved 899 out of 904 problems, more than any of the other systems.

Both SHOP and SHOP2 are open-source software, and may be downloaded at ⟨http://www.cs.umd.edu/projects/shop⟩. SHOP is available in both Common Lisp and Java. SHOP2 is only available in Common Lisp, but it includes an interface for interoperating with programs written in other languages, and we are currently implementing a Java version.

3.3 CAT

In planning a course of action (i.e., a plan to achieve a desired objective or objectives), quick and accurate decision making is a very important task and it is very hard. A major source of difficulty is how to deal with uncertainty. This uncertainty has many sources, but perhaps the biggest one is the uncertain relationship between causes and effects. For example:

- *At a tactical level, sorties are flown against a series of bridges to prevent the enemy ground forces from crossing the river. The sorties are intended to prevent the crossing. What is the probability that they will?*
- *At a strategic level, the destruction of the Taliban Army was intended ultimately to reduce world-wide terrorism. Did it?*

Such uncertainties are compounded by the size and complexity of most military plans—for example, a causal model of Operation Deny Freedom, built by the actual planners, contains over 300 uncertain events interrelated by cause and effect. Moreover, there are often significant delays between cause and effect, and effects may persist for only limited amounts of time: a destroyed bridge can be rebuilt or bypassed. This makes it exceedingly difficult to forecast the possible effects of a military operation.

This section describes the approach we have developed to help analyze this uncertainty in order to generate effective plans. The basis for our approach is the Air Force Research Laboratory's (AFRL's) Causal Analysis Tool (CAT), which is a tool for representing and analyzing causal networks similar to Bayesian networks. From this representation, CAT can compute the probability that any given plan (i.e., any chosen combination of actionable items) will achieve the desired objectives.

A major technical difficulty is how to overcome combinatorial blowup during the planning process. If there are n different actionable items, then there are potentially 2^n different plans, making it infeasible for the user to ask CAT to analyze each one. Our approach exploits the conditional-independence relationships within a causal network in order to overcome this combinatorial blowup. In doing so, it quickly computes upper and lower bounds on the probabilities of success associated with a partial plan, and uses these bounds to recommend which actions the user should include in the plan in order to get a complete course of action. This provides an exponential reduction in amount of time needed to find a complete plan. In our experimental evaluation, our approach generated recommendations that resulted in plans that have the highest probability of success in just a few minutes, demonstrating its effectiveness.

3.3.1 Background: Causal Analysis Tool (CAT)

Causal Analysis Tool (CAT) is a system developed by the Air Force Research Laboratory (AFRL) for being use in creating, modifying and analyzing causal models. CAT is a development tool that is currently in prototype stage and it has not been deployed in any sort of active use yet. However, to the best of our knowledge, several strategic-level organizations within the US Air Force are testing CAT and giving positive feedback about it.

Probability Analysis in CAT

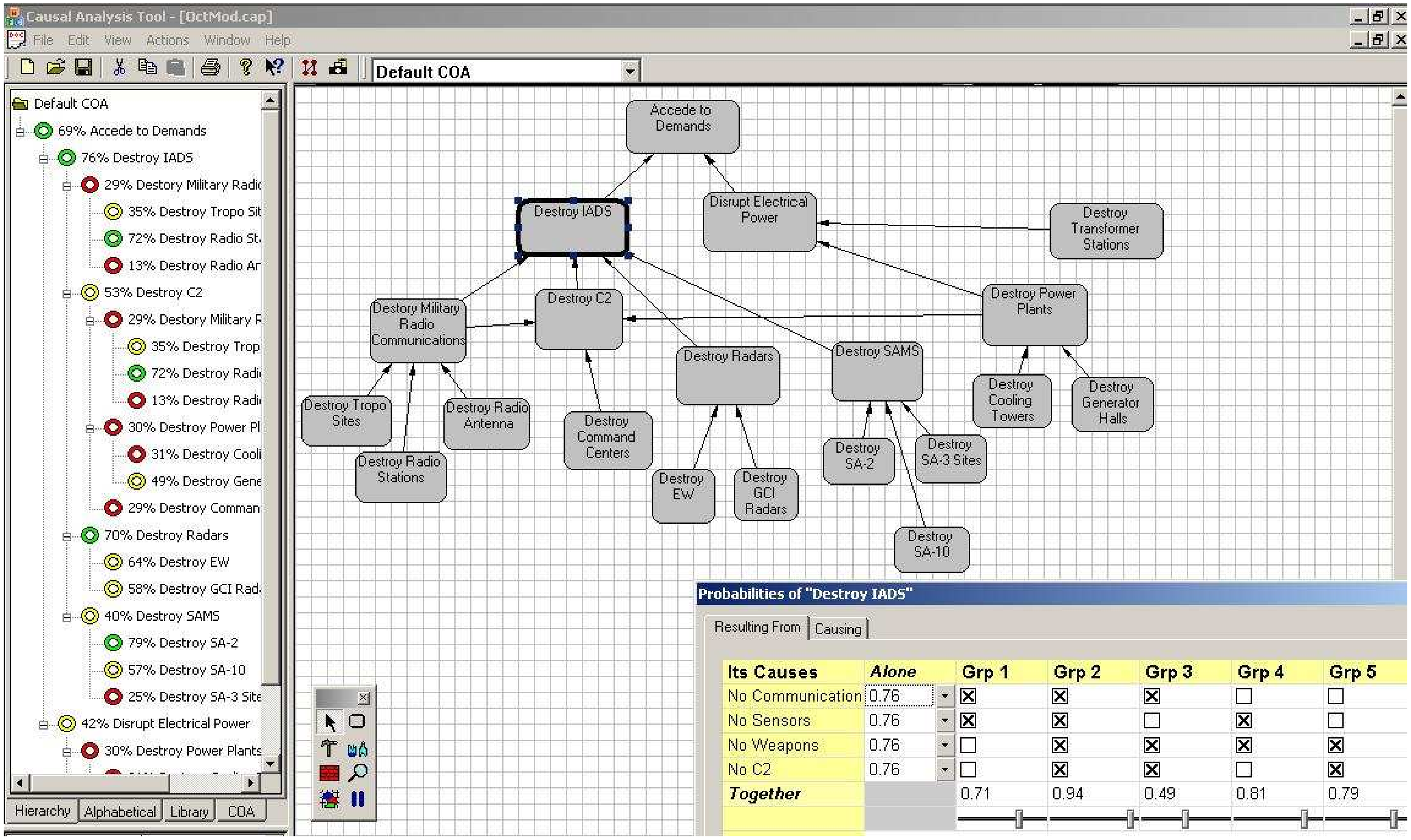
The basic function of CAT is to propagate local estimates of uncertainty throughout large models. Its most basic output is the probability, as a function of time, that particular events will be true. Below, we give a brief summary of CAT; for detailed information on the technology that CAT uses, see [44, 45, 46].

Probability analysis in CAT is based on the use of causal models; CAT provides tools to either construct a causal model or load a previously constructed causal model from a file. CAT's causal models are similar to Bayesian Networks (and CAT compiles them into Bayesian Networks in order to do its analysis). However, CAT's causal models incorporate several extensions in order to make Bayesian causal modeling available to users who do not have specialized probability training, and allow sophisticated incremental improvement of these models when more time is available.

In CAT, a *causal model* is a directed graph (e.g., see Figure 3.7 on the next page) in which each node represents an event that may or may not occur. There are three different kinds of events: *actionable items*, which are actions that we may choose whether or not to perform, *goals* that we may wish to achieve, and other intermediate events that are neither actionable items nor goals. The edges (which are called *mechanisms*) represent causal and inhibitory relationships between events. A mechanism $e_1 \mapsto e_2$ between events e_1 and e_2 is *causal* if the occurrence of e_1 increases e_2's probability of occurrence, and it is *inhibitory* if the occurrence of e_1 reduces e_2's probability of occurrence. Associated with each mechanism is a number between 0 and 1 to indicate the probability with which e_1 causes or inhibits e_2. These numbers are probabilities of causation or inhibition rather than the conditional probabilities used in Bayesian networks—but they can be translated into the latter, and CAT does such a translation in order to perform its calculations.

In a causal model, the user can specify a number of probabilities by filling in the probability tables for each event in the causal model. For example, Figure 3.7 shows a set of user-specifed causal probabilities for the event ”Destroy IADS” in that model. These probabilities tell us that each of the mechanisms ”No Communications”, ”No Sensors”, ”No Weapons”, and ”No C2” will cause this event alone with probability 0.76. The user can also specify causal probabilities for the event ”Destroy IADS” given various groups of its causes by using the ”group” check-boxes.

Furthermore, each event in a causal model is associated with a special type of probability, called the *leak probability* for that event. Intuitively, an event's leak probability specifies the probability that the event will occur even when none of its causes occurs in the world. In other words, a leak probability specifies the causes of an event that are not specified explicitly in the given causal model. Leak probabilities allow CAT users to work with incomplete causal models with unknown events and still be able to reason and compute the probabilities of the events already in the causal model.



Default COA
- 69% Accede to Demands
  - 76% Destroy IADS
    - 29% Destory Military Radic
      - 35% Destroy Tropo Sit
      - 72% Destroy Radio St.
      - 13% Destroy Radio Ar
    - 53% Destroy C2
      - 29% Destory Military R
        - 35% Destroy Trop
        - 72% Destroy Radi
        - 13% Destroy Radi
      - 30% Destroy Power Pl
        - 31% Destroy Cooli
        - 49% Destroy Gene
      - 29% Destroy Comman
    - 70% Destroy Radars
      - 64% Destroy EW
      - 58% Destroy GCI Rad
    - 40% Destroy SAMS
      - 79% Destroy SA-2
      - 57% Destroy SA-10
      - 25% Destroy SA-3 Site
  - 42% Disrupt Electrical Power
    - 30% Destroy Power Plants

Probabilities of "Destroy IADS"

Its Causes	Alone	Grp 1	Grp 2	Grp 3	Grp 4	Grp 5
No Communication	0.76	☒	☒	☒	☐	☐
No Sensors	0.76	☒	☒	☐	☒	☐
No Weapons	0.76	☐	☒	☒	☒	☒
No C2	0.76	☐	☒	☒	☐	☒
Together		0.71	0.94	0.49	0.81	0.79

Figure 3.7: An abstract causal model named "Operation OctMod" which represents the plan that the international coalition used against Milosevic in the Bosnia-Herzegovina war. The window on the right-hand side of the screen shows a portion of the probability table stored in the highlighted node. The actionable items are the twelve nodes at the bottom of the network that have no predecessors.

To calculate the probabilities of occurrence for the events and mechanisms of a causal model M, CAT first compiles M into a Bayesian Network, say $B(M)$, such that the event and mechanisms in M correspond to the nodes of $B(M)$. CAT computes two different conditional probability tables (CPTs) for each node n in $B(M)$; namely, a ***Causal CPT*** and a ***Inhibitory CPT***. These conditional probability tables model the causal and inhibitory relationships among the events and mechanisms of a causal model, as described above. They are both computed using the *Recursive Noisy-OR (RNOR) rule* reported in [46]. The RNOR rule is a generalization of the traditional *Noisy-OR* rule [59], which is widely used for computing the probability of an event, given the conditional probabilities that describe the dependencies between that event and each of its predecessors. The RNOR rule allows for modeling and reasoning about complex dependencies between events of a given causal model, which cannot be captured by the Noisy-OR rule. For details on the RNOR rule, see [46].

Having computed the special conditional probability tables described above, CAT performs a variant of ***probabilistic logic sampling*** [32] over the compiled Bayesian Network $B(M)$, in which it repeatedly simulates the occurrence (or nonoccurrence) of the nodes in $B(M)$.[4] In each simulation run, CAT decides whether an event (i.e., a node in $B(M)$) n occurs with the probability computed by the following formula

$$(1.0 - InhibitoryCPT(n, inhibitors(n))) \times [1.0 - (1.0 - CausalCPT(n, causes(n)))(1.0 - Leak(n))],$$

where $Leak(n)$ is the leak probability associated with the node n, and $CausalCPT$ and $InhibitoryCPT$ denote the causal and inhibitory conditional probability tables computed by CAT for the node n during the compilation phase. $causes(n)$ and $inhibitors(n)$ are the sets of predecessors of n in $B(M)$ such that $causes(n)$ denotes the set of nodes in $B(M)$ whose occurrence increases the probability of occurrence for n, and $inhibitors(n)$ denote the set of nodes whose occurrence decreases the probability of occurrence for n.

The formula given above specifies the following probabilistic-reasoning behavior: if a node n had no predecessors in $B(M)$ that inhibit the occurrence of n, then we would want n to occur as a result of its causal dependencies specified in $CausalCPT(n, causes(n))$, and/or as a result of unmodeled external factors with probability $Leak(n)$. However, if n has inhibiting predecessors, then the probability of the occurrence of n due to its causes and its leak probability value may be reduced with the probability specified in $InhibitoryCPT(n, inhibitors(n))$. Thus, in each simulation, n will occur with the probability computed with the formula above, given the occurrences and non-occurrences of each of its predecessors in that simulation run.

In each simulation run, CAT starts with the nodes in $B(M)$ that has no predecessors. For such nodes, the above formula specifies the "a priori" probabilities given as input by the user. The simulation progresses by iteratively considering each node n in $B(M)$ such that the occurrence or non-occurrence of all of the predecessors of n is already probabilistically simulated in this particular run. This way, when CAT considers to simulate the occurrence or nonoccurrence of a node n in a run, it always knows whether the predecessors of n occurred or not in that particular simulation run. In other words, CAT always knows whether the nodes in $causes(n)$ and $inhibitors(n)$ are occured or not in that particular simulation run, when it considers the node n.

CAT runs its simulation repeatedly, for as long as the user wants. As it does so, it keeps statistics on how frequently each node occurs. It uses these statistics to compute an estimate of the probability of occurrence for every event in the original causal network M. CAT displays these estimates to the user as shown in the left-hand pane of Fig. 3.7. As CAT runs more and more simulations, the estimates of each such probability get progressively more accurate, and CAT updates its display accordingly. The user may stop running simulations whenever he/she feels that the estimates have become sufficiently accurate.

[4]The reason why CAT uses probabilistic logic sampling is because of the way in which CAT reasons about time and scheduling; the details are beyond the scope of this paper.

Planning using CAT

In CAT, planning takes place as an iterative and interactive process in which users repeatedly do the following: (1) they make decisions about some actionable items to include and/or exclude, (2) they use CAT to obtain an estimate of the probability of achieving the goal,[5] and (3) they revise these decisions based on their experience and intuition.

Users may need to try many combinations of actionable items in order to generate the plan that has the highest probability of achieving the goal. This plan is not necessarily the one that includes all possible actionable items: if the causal model contains inhibitory mechanisms, then some actionable items may reduce the probability of achieving the goal. In order to find the plan that maximizes the probability of achieving the goal, in the worst case a user may need to create and analyze exponentially many alternative plans. For example, if there are n actionable items, then there are 2^n different possible combinations of the actionable items, i.e., 2^n different plans. Since the causal models can be quite large and complex, and since the planning often needs to be done in a very limited amount of time under stressful conditions, it clearly is not feasible for the user to generate and examine all of these plans.

As an example, if $n = 22$ then there are 2^{22} different possible plans. Suppose CAT takes 10 seconds to analyze each plan (this assumption is rather optimistic: if the network is sufficiently large, CAT might take minutes or even hours). Then the total time needed to analyze all of the plans is approximately 11,651 hours, or more than 485 days. Clearly, this is not acceptable.

3.3.2 Our Approach

We have developed a way to overcome the exponential blowup described above. Our approach involves modifying CAT so that it can represent and reason about *partial plans* in which the user has made yes-or-no decisions for some of the actionable items and the others remain *undecided*. This enables the users to carry out the following *iterative plan-development process*: the user begins with a partial plan in which all actionable items are undecided, and gradually makes decisions about more and more of the items until no undecided items remain.

By using our technique, we can give the following feedback to the user at each iteration of the planning process: (1) upper and lower bounds on the probabilities of success that can be attained with the current partial plan, and (2) a recommendation for what choices to make next in order to achieve a complete plan. The following subsections describe how we compute the upper and lower bounds, and how we use these bounds to recommend which actionable items to include or exclude next.

Upper and Lower Bounds

We now discuss how to compute lower and upper bounds $P_{\min}(e)$ and $P_{\max}(e)$ on the probability of each event in a causal model M.

It is simple to put lower and upper bounds on the probabilities of the actionable items. Suppose the set of actionable items is $A = \{a_1, \ldots, a_n\}$, and suppose the user has already chosen some set of actions $D^+ \subseteq A$ to include in the plan and some subset $D^- \subseteq A$ to exclude from the plan, so that the current partial plan is $D = D^+ \cup \{\neg a_i : a_i \in D^-\}$. Then for each $a_i \in D^+$, $P_{\min}(a_i) = P_{\max}(a_i) = 1$; and for each $a_i \in D^-$, $P_{\min}(a_i) = P_{\max}(a_i) = 0$. For each $a_i \in A \setminus (D^+ \cup D^-)$, the user has not yet decided whether to include a_i in the plan, so the tightest lower and upper bounds we can place on $P(a_i)$ are $P_{\min}(a_i) = 0$ and $P_{\max}(a_i) = 1$.

[5]For simplicity, in this paper we assume that there is just one goal g. Situations in which there are several goals $g_1, \ldots, g_k$ can sometimes be modeled by adding a new node g whose causes are $g_1, \ldots, g_k$.

Given the probabilities $\{P_{\min}(a_i), P_{\max}(a_i)\}_{i=1}^n$, we want to compute $P_{\min}(e)$ and $P_{\max}(e)$ for every event in M that is not an actionable item. One way would be the brute-force approach: run CAT's probability analysis on M repeatedly, once for every combination of probabilities $\{P(a_i) \in \{0,1\} : a_i \in A \setminus D\}$. However, this approach incurs the same kind of exponential blowup that we discussed earlier, because it requires doing the probability analysis 2^{n-m} times, where $n = |A|$ and $m = |D|$. As we now describe, a quicker computation can be done by taking advantage of conditional independence among the events in M.

During CAT's simulations, the occurrence or non-occurrence of an event e in the Bayesian network $B(M)$ is represented by a boolean random variable $x(e) \in \{0,1\}$. During each simulation run, the probability that CAT assigns $x(e) = 1$ is $P(e)$. In our modified version of CAT, the simulation procedure instead uses *two* random variables $x_{\min}(e)$ and $x_{\max}(e)$ for each event e. Our simulation assigns $x_{\min}(e) = 1$ with a probability that is a lower bound on $P(e)$, and it assigns $x_{\max}(e) = 1$ with a probability that is an upper bound on $P(e)$. This is done as follows.

If e is an actionable item, then there are three cases:

- If the user has chosen to include in the plan, we assign $x_{\min}(e) = x_{\max}(e) = 1$.
- If the user has chosen not to include in the plan, we assign $x_{\min}(e) = x_{\max}(e) = 0$.
- Otherwise we assign $x_{\min}(e) = 0$ and $x_{\max}(e) = 1$.

If e is not an actionable item, then let $e_1, e_2, \ldots, e_b$ be all of the nodes that may affect e, i.e., $e_1, e_2, \ldots, e_b$ are the predecessors of e. Suppose the simulation has progressed far enough to assign values to $x_{\min}(e_i)$ and $x_{\max}(e_i)$ for $i = 1, \ldots, b$. From conditional independence, it follows that $P(e)$ depends only on $e_1, \ldots, e_b$. Thus, the set of possible probabilities for e is

$$\begin{aligned} S = \{P(e|x(e_1), x(e_2), \ldots, x(e_b)) :& \\ & x(e_1) \in \{x_{\min}(e_1), x_{\max}(e_1)\}, \\ & x(e_2) \in \{x_{\min}(e_2), x_{\max}(e_2)\}, \\ & \ldots, \\ & x(e_b) \in \{x_{\min}(e_b), x_{\max}(e_b)\}\}. \end{aligned}$$

Then the simulation assigns $x_{\min}(e) = 1$ with probability $\min(S)$, and assigns $x_{\max}(e) = 1$ with probability $\max(S)$.

Providing Feedback and Recommendations

Like the original version of CAT, the modified version can keep running simulations for as long as the user wishes. Suppose that the user has made some set of yes-or-no decisions D. For each node e, let $P^k_{\min}(e|D)$ and $P^k_{\max}(e|D)$ be the average values of $x_{\min}(e)$ and $x_{\max}(e)$ over a set of k simulation runs. Our modified version of CAT displays these averages to the user as shown in the left-hand panes of Figures 4.4, 4.5, and 4.6. As the number of runs increases, $P^k_{\min}(e|D)$ and $P^k_{\max}(e|D)$ converge to lower and upper bounds on $P(e|D)$.

Our modified version of CAT uses a hill-climbing approach to provide recommendations for additional actions to include in D^+ and D^-. Suppose g is some *goal event* whose probability the user wants to maximize. In addition to computing $P^k_{\min}(g|D)$ and $P^k_{\max}(g|D)$ as described above, our modified version of CAT also computes $P^k_{\min}(g|D, a_i)$ and $P^k_{\min}(g|D, \neg a_i)$ for every $a_i \in A \setminus (D^- \cup D^+)$. Let

$$P^* = \max \bigcup_i \{P^k_{\min}(g|D, a_i), P^k_{\min}(g|D, \neg a_i)\}.$$

Then P^* is the largest amount by which $P_{\min}(g|D)$ can increase if the user makes a yes-or-no decision about one of the undecided actions. Either there is an a_i such that $P^k_{\min}(g|D, a_i) = P^*$, in which case our modified version of CAT will recommend including a_i in the plan, or else there is an a_i such that $P^k_{\min}(g|D, \neg a_i) = P^*$, in which case our modified version of CAT will recommend excluding a_i from the plan.

Computation Time

The total computation time required by this technique is no greater than the time needed for $n2^b$ calls to the original version of CAT, where b is the maximum number of predecessors of each node and n is the number of actionable items. This is a substantial improvement over 2^n, because b normally remains small even in very large networks. For example, in the OctMod example of Fig. 3.7, no node has more than four predecessors. Furthermore, if most nodes have fewer than b predecessors (as is true in the OctMod example), then the total computation time will be substantially less than $n2^b$.

For example, let us suppose that we have causal model in which the maximum number of predecessors of each node is $b = 4$, the number of actionable items is $n = 25$ and three of the actionable items has been already decided — i.e., we have $m = 3$. Furthermore, suppose again that CAT needs 10 seconds each time it analyzes the causal network. Then the total time needed for us to get the complete plan is less than 70 minutes. This is substantially better than the 485 days required by the brute-force approach!

Chapter 4

Results and Discussion

The first three sections describe the results for HICAP, SHOP and SHOP2, and CAT, respectively. The fourth section is a survey of related work.

4.1 Results for HICAP

The following sections give theoretical results on the correctness of HICAP's SiN algorithm, and describe an empirical analysis that demonstrates the impact of the preferences on plan quality.

4.1.1 Correctness of SiN

In this section we will assume that SiN performs ordered task decomposition. That is, we assume that all tasks are totally ordered and at each iteration, when refining a set of tasks T', SiN will start by decomposing the first task in T'.

If I is an incomplete domain description and B is a case base (i.e., a set of cases), then a domain description D is consistent with $I \cup B$ iff (1) every method and operator in I is an instance of a method or operator in D and (2) for every case $C = (h, P, ST, Q)$ in B, there is a method $M = (h', P', ST')$ in D such that h, P, and ST are instances of h', P' and ST' respectively. Although many different domain theories might be consistent with $I \cup B$, in general we will not know which of these is the one that produced I and B. However, SiN is correct in the sense that, if it succeeds in outputting a plan, then that plan could have been generated by JSHOP using any domain description consistent with $I \cup B$.

Theorem 1 (Correctness of SiN) *Let T be a collection of tasks, S be an initial state, I be an incomplete domain description, and B be a case base, and let SiN(T, S, I, B) represent the invocation of SiN with those items as inputs. Suppose that SiN performs ordered task decomposition. Then:*

- *If SiN(T, S, I, B) returns a plan π, then for every domain description D consistent with $I \cup B$, π is a solution plan for the planning problem (T, S, D).*
- *If SiN(T, S, I, B) cannot find a plan, then there is a domain description D consistent with $I \cup B$ such that no solution plan exists for (T, S, D).*

The proof is done by induction on the number of iterations of the SiN algorithm. The proof shows that each SiN task decomposition in $(T, S, I \cup B)$ corresponds to a JSHOP task decomposition in (T, S, D). This is sufficient to prove correctness, because of the correctness of JSHOP's planning algorithm [58].

This proposition suggests that cases in SiN supply two kinds of knowledge: first, they provide control knowledge, similar to the knowledge encoded in cases using derivational replay when a complete domain description is available [71, 33]. Because cases are instances of methods, applying a case is comparable to a replay step in which the method selected to decompose a task is the one in the case's derivational trace. The main difference is that, while cases in replay systems correspond to a complete derivational trace, cases in SiN correspond to a single step in the derivational trace. Second, cases in SiN augment the domain description and, thus, provide domain knowledge as do cases in many case-based planners (e.g., [30]).

Imperfect World Information

SiN uses NaCoDAE to dynamically elicit the world state, which involves obtaining the user's preferences. Depending on the user's answers, cases will get re-ranked. When solving a task, the user can choose any of the cases, independent of their ranking, provided that all their preconditions are met. The preferences play a pivotal role in determining plan quality due to the absence of a complete domain description.

Consider the following two simplified cases:

Case 1:

Head: selectTransport(ISB,evacuation-site)
Preconditions: HelosAvailable(ISB)
Questions-Answer pairs: Weather conditions? Fine
Subtasks: Transport(ISB,evacuation-site,HELOS)

Case 2:

Head: selectTransport(ISB,evacuation-site)
Preconditions: groundTransportAvailable(ISB)
Questions-Answer pairs:
Weather conditions? Rainy
Imminent danger to evacuees? No
Subtasks: Transport(ISB,evacuation-site,GroundTransport)

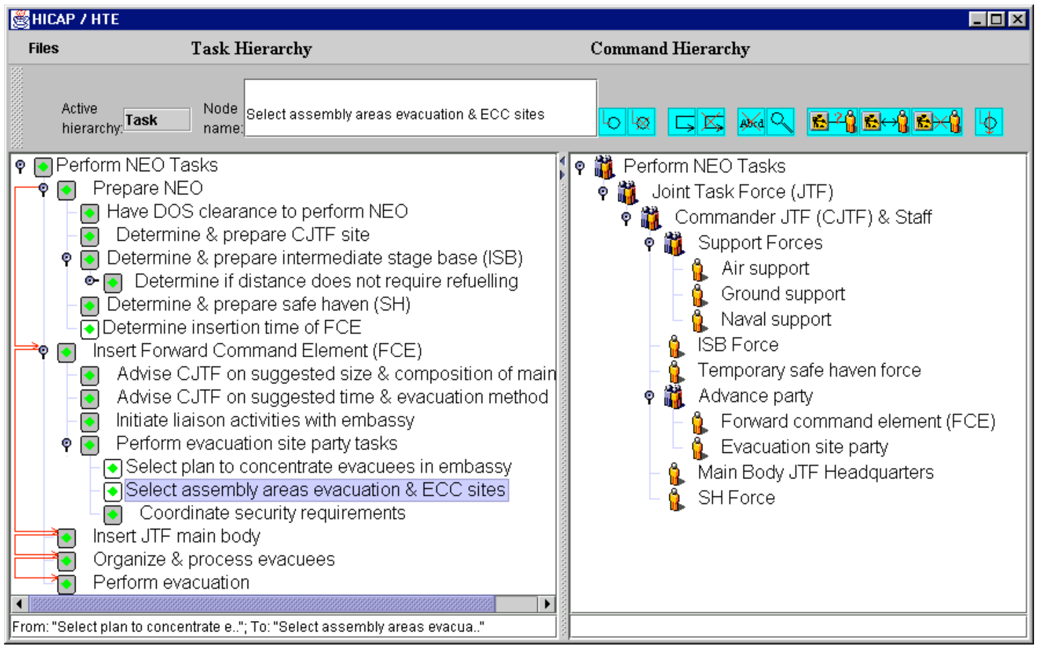
These cases both concern the selection of transportation means between an ISB and the site to be evacuated. The first case suggests using helicopters provided that they are available at the ISB. The second one suggests using ground transportation provided that the corresponding transportation means are available at the ISB. If the two cases are applicable, because both preconditions are met, the answers given by the user will determine a preference between them. For example if the weather is rainy and there is no immediate danger for the evacuees, NaCoDAE would suggest the second case. The rationale behind this is that flying in rainy conditions is risky. Thus, selecting ground transportation would be a better choice.

4.1.2 Example

During a typical planning episode, the user views the top-level tasks first, revising them or their assignments if necessary. He/she may choose to decompose any of the tasks and view their decomposition. Figure 4.1 shows an intermediate stage during this process. The user has selected the task "Select assembly areas for evacuation & ECC (Evacuation Control Center) sites." Thus, the left-hand pane highlights this task, and the right-hand pane highlights the name of the group responsible for performing it.

Several alternative methods can be considered for decomposing the "select assembly areas" task. When the planner selects this task, HICAP starts NaCoDAE, which displays the alternatives along with two questions to help distinguish which one is the best match (see Figure 4.2(a)).

In Figure 4.2(b), the user has answered one of the questions and this has yielded a perfect match to one of the cases for the "select assembly areas" task. Suppose that the user selects this case to decompose this task. Figure 4.3 shows the result of this decomposition; two new subtasks are displayed that correspond to



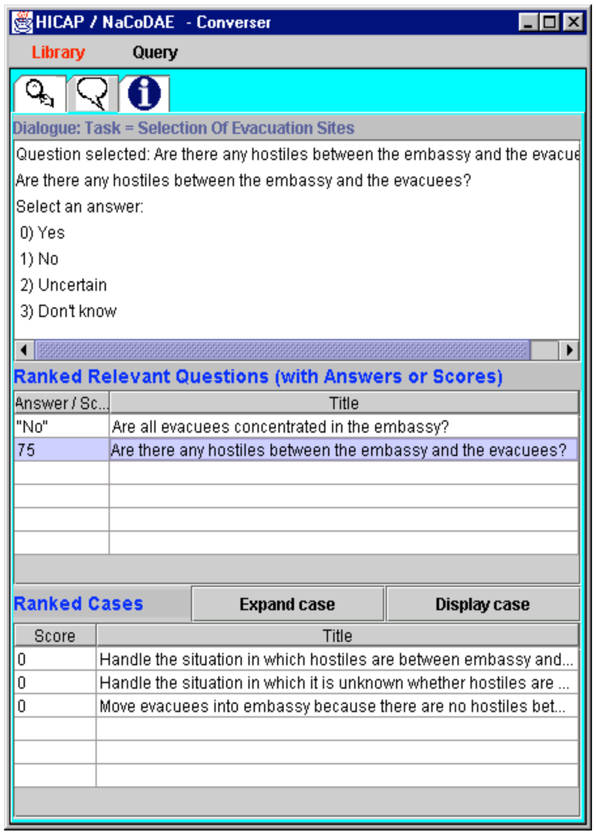
Select assembly areas evacuation & ECC sites

- Perform NEO Tasks
  - Prepare NEO
    - Have DOS clearance to perform NEO
    - Determine & prepare CJTF site
    - Determine & prepare intermediate stage base (ISB)
      - Determine if distance does not require refuelling
    - Determine & prepare safe haven (SH)
    - Determine insertion time of FCE
  - Insert Forward Command Element (FCE)
    - Advise CJTF on suggested size & composition of main
    - Advise CJTF on suggested time & evacuation method
    - Initiate liaison activities with embassy
    - Perform evacuation site party tasks
      - Select plan to concentrate evacuees in embassy
      - Select assembly areas evacuation & ECC sites
      - Coordinate security requirements
  - Insert JTF main body
  - Organize & process evacuees
  - Perform evacuation

From: "Select plan to concentrate e.."; To: "Select assembly areas evacua.."

- Perform NEO Tasks
  - Joint Task Force (JTF)
    - Commander JTF (CJTF) & Staff
      - Support Forces
        - Air support
        - Ground support
        - Naval support
      - ISB Force
      - Temporary safe haven force
      - Advance party
        - Forward command element (FCE)
        - Evacuation site party
      - Main Body JTF Headquarters
      - SH Force

Figure 4.1: A snapshot of HTE's interface, displaying tasks (left) and the JTF's hierarchical organization (right). Arrows denote ordering constraints.

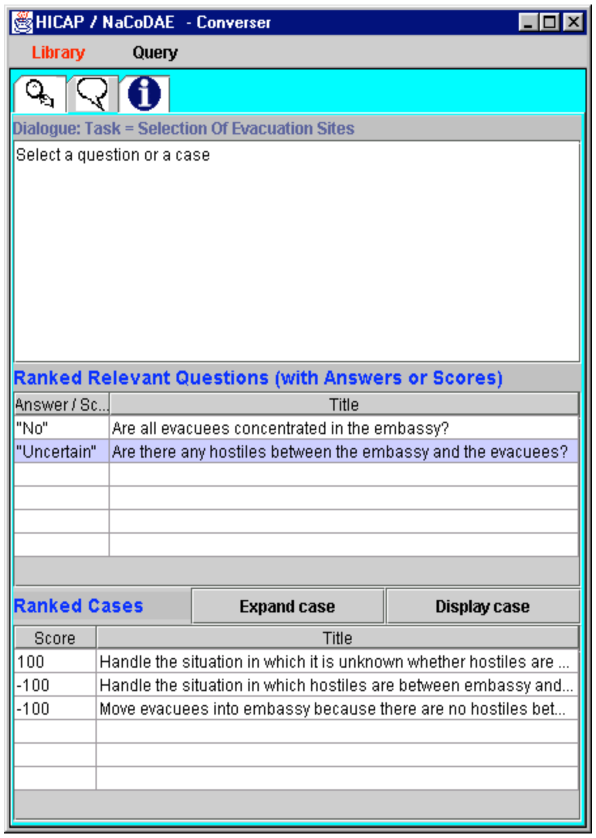

HICAP / NaCoDAE - Converser
Library
Query
Ranked Relevant Questions (with Answers or Scores)
Ranked Cases
Expand case
Display case

Dialogue: Task = Selection Of Evacuation Sites

Question selected: Are there any hostiles between the embassy and the evacue
Are there any hostiles between the embassy and the evacuees?
Select an answer:
0) Yes
1) No
2) Uncertain
3) Don't know

Answer / Sc...	Title
"No"	Are all evacuees concentrated in the embassy?
75	Are there any hostiles between the embassy and the evacuees?

Score	Title
0	Handle the situation in which hostiles are between embassy and...
0	Handle the situation in which it is unknown whether hostiles are ...
0	Move evacuees into embassy because there are no hostiles bet...


HICAP / NaCoDAE - Converser
Library
Query
Ranked Relevant Questions (with Answers or Scores)
Ranked Cases
Expand case
Display case

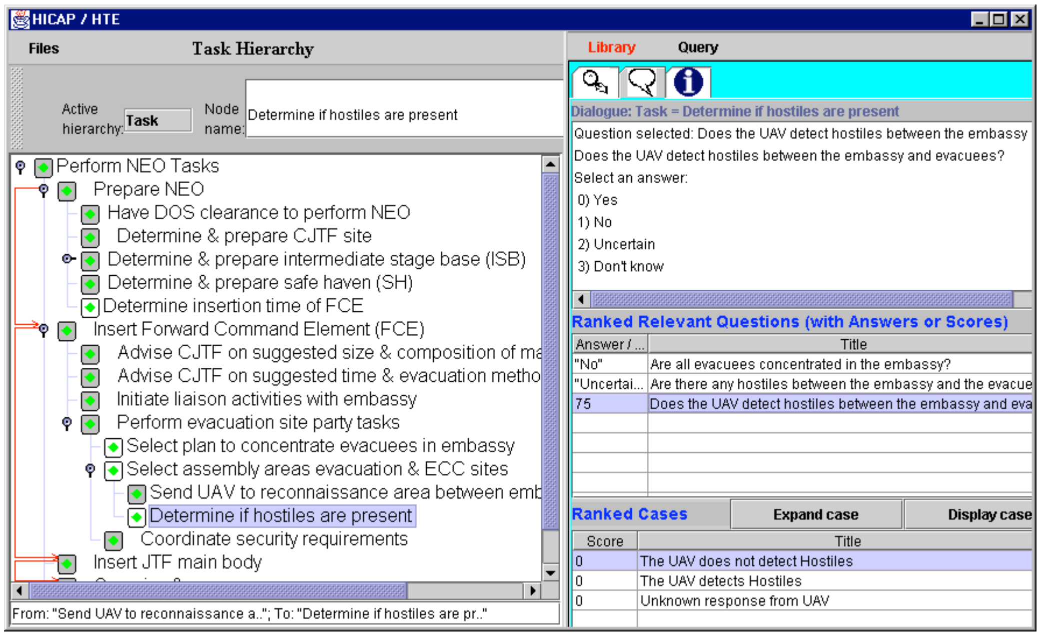
Dialogue: Task = Selection Of Evacuation Sites

Select a question or a case

Answer / Sc...	Title
"No"	Are all evacuees concentrated in the embassy?
"Uncertain"	Are there any hostiles between the embassy and the evacuees?

Score	Title
100	Handle the situation in which it is unknown whether hostiles are ...
-100	Handle the situation in which hostiles are between embassy and...
-100	Move evacuees into embassy because there are no hostiles bet...

Figure 4.2: Snapshots of NaCoDAE/HTN's interface before and after a question has been answsered. In each, the top window displays advice on what to do next and, when the user answers a question, lists the possible answers. The lower windows display the questions and cases, respectively.


HICAP / HTE

Files Task Hierarchy

Active hierarchy: Task Node name: Determine if hostiles are present

- Perform NEO Tasks
  - Prepare NEO
    - Have DOS clearance to perform NEO
    - Determine & prepare CJTF site
    - Determine & prepare intermediate stage base (ISB)
    - Determine & prepare safe haven (SH)
    - Determine insertion time of FCE
  - Insert Forward Command Element (FCE)
    - Advise CJTF on suggested size & composition of ma
    - Advise CJTF on suggested time & evacuation metho
    - Initiate liaison activities with embassy
    - Perform evacuation site party tasks
      - Select plan to concentrate evacuees in embassy
      - Select assembly areas evacuation & ECC sites
        - Send UAV to reconnaissance area between emb
        - Determine if hostiles are present
      - Coordinate security requirements
  - Insert JTF main body

From: "Send UAV to reconnaissance a.."; To: "Determine if hostiles are pr.."

Library Query

Dialogue: Task = Determine if hostiles are present

Question selected: Does the UAV detect hostiles between the embassy

Does the UAV detect hostiles between the embassy and evacuees?

Select an answer:

0) Yes

1) No

2) Uncertain

3) Don't know

Ranked Relevant Questions (with Answers or Scores)

Answer / ...	Title
"No"	Are all evacuees concentrated in the embassy?
"Uncertai...	Are there any hostiles between the embassy and the evacue
75	Does the UAV detect hostiles between the embassy and eva

Ranked Cases Expand case Display case

Score	Title
0	The UAV does not detect Hostiles
0	The UAV detects Hostiles
0	Unknown response from UAV

Figure 4.3: HICAP's interface after decomposing the "select assembly areas" task.

this case's decomposition network. Interaction can continue in a similar manner until all of the operational elements of the plan have been elaborated.

4.2 Results for SHOP and SHOP2

SHOP and SHOP2 are available as freeware and have been downloaded thousands of times,[1] and have developed a significant user base, including users from government laboratories, industries, and universities. Based on information given to us by some of the users on our mailing list, here are descriptions of a few of the projects in which SHOP and SHOP2 have been used.

4.2.1 Projects in US Government Laboratories

Noncombatant Evacuation Planning (Naval Research Laboratory, Washington, DC)

The planning component of the HICAP system (see Section 3.1) includes both generative and case-based planning. The generative component is provided by SHOP. SHOP and the case-based component, NaCoDAE are integrated quite tightly: each is capable of using the other to decompose tasks into subtasks.

Evaluating Terrorist Threats (Naval Research Laboratory, Washington, DC)

The purpose of NRL's AHEAD project is to help intelligence analysts understand and evaluate hypotheses about terrorist threats. Given a hypothesis, the AHEAD system uses analogical retrieval to obtain a model of the hostile activity most closely related to the hypothesis. This model is encoded as an HTN domain description for SHOP2 in which individual actions are annotated with additional explanatory information about their function. AHEAD invokes SHOP2 using this domain description to

[1]As of June 24, the log of downloads from our web site shows 1783 downloads, but this does not include the number of times that users have downloaded directly from our ftp server rather than going through our web site. We imagine the total number of downloads is above 2000.

produce a plan that is compatible with the hypothesis. As each operator is added to the plan, SHOP2 queries an external evidence database to determine whether the evidence is consistent with that operator. When the evidence is consistent with the operator, AHEAD generates an argument in favor of the hypothesis; when the evidence is inconsistent, AHEAD generates a counterargument. The resulting structured argument is presented in a browsable user interface. HTN planning is particularly well-suited to this process because HTNs organize behavior into meaningful components at multiple levels of abstraction thus enabling coherent, structured argumentation. For more information about AHEAD, see ⟨http://www.nrl.navy.mil/aic/iss/ida/projects/ahead/AHEAD.php⟩.

Software Systems Integration (NIST, Gaithersburg, MD)

NIST (National Institute of Standards and Technology) is using SHOP2 in a project whose goal is to automate tasks of software systems integration. So far, the particular example they have used is based on General Motors' ebXML-based "bulk rental car buying" interfaces. These interfaces allow a buyer to search for cars of a particular make, model and year, and purchase them. But if a buyer wants to get a summary of various cars at various locations (e.g., in order to minimize costs), the interface makes it difficult to do this. NIST's code reads the seller's ebXML BPSS (business process specification schema) and produces a SHOP2 planning problem in which SHOP determines the sequence of transactions against the seller's interfaces to achieve the buyer's objective. For further information about the project, see ⟨http://www.mel.nist.gov/proj/mee.htm⟩.

4.2.2 Projects in Other Government Laboratories

Fighting Forest Fires (LAAS/CNRS, Toulouse, France)

The European Union's COMETS project focuses on the development of unmanned aerial vehicle (UAV) control techniques for detection and monitoring of forest fires. As part of this project, researchers at LAAS, a government research laboratory in Toulouse, France, are developing a distributed architecture in which each UAV will contain a generic "decisional node" consisting of a supervisor and a planner.

Within each decisional node, they are using SHOP2 as the symbolic planner: they exploit SHOP2's forward-chaining capability to integrate its planning activity with specialized software for estimating the costs, time, etc., for basic UAV operations. In order to perform temporal reasoning, they are using the same time-stamping technique we developed for temporal planning with SHOP2 in the 2002 International Planning Competition [57]. The researchers anticipate that they soon will have simulation results and will be able to run experiments using LAAS's blimp, *Karma*. More information about the project is available at ⟨http://www.comets-uavs.org⟩.

4.2.3 Industry Projects

Controlling Multiple UAVs (SIFT, Minneapolis, MN)

SIFT, LLC is using a modified version of the SHOP2 planner in a UAV control system in their PVACS project, with funding from DARPA through an SBIR contract. SIFT's "Playbook" control system allows time-pressured users, who are not UAV operators, to request reconnaissance missions using high-level tasking commands, modeled on the way people delegate tasks to human subordinates. The SIFT Playbook supports interactions through both PDA and desktop/laptop interfaces. The Playbook translates users' brief, general commands into very specific control actions suitable for execution. The Playbook's Executive provides high-level closed-loop monitoring and implementation of the Playbook's plans, controlling multiple UAVs through the Variable Autonomy Control System (VACS) Ground Control Station (GCS), developed by Geneva Aerospace, Inc. The Playbook currently operates these UAVs in a high-fidelity simulation environment, but the interface it uses to control the simulated UAVs through the VACS GCS is the same as the one used to direct VACS UAVs in real flight operations.

The modified SHOP2 planner plays a key role in SIFT's Playbook, translating the user's high level task specifications into a sequence of commands that can be executed by UAVs. SIFT's plan library contains tasks for multiple reconnaissance missions, for both rotorcraft and fixed-wing UAVs. Robert Goldman at SIFT has developed an augmented version of SHOP2 that generates temporal plans including durative actions, and provides more knowledge-engineering and debugging support. For further information, see ⟨http://www.sift.info/English/projects/PVACS.ppt⟩.

Evaluation of Enemy Threats (Lockheed Martin ATL, Cherry Hill, NJ)

Lockheed Martin Advanced Technology Laboratories, in collaboration with the Army Research Laboratory, is using SHOP in a project that attempts to evaluate possible enemy threats. They are using SHOP to decompose higher level tasks such as 'attack blue-convoy' into sequences of operations such as 'move red-tank1 to location2, . . . , fire red-tank1 at blue-convoy.' Due to their confidentiality restrictions, they were unable to tell us any further details.

Location-Based Services (Sony Electronics, San Jose, CA)

Sony Electronics Incorporated has used SHOP in a project aimed at developing mobile GIS devices to help people plan errands that take them to different geographical locations. Due to Sony's confidentiality requirements, they were unable to tell us any further details.

Material Selection for Manufacturing (Infocraft Ltd., Sri Lanka)

Infocraft Ltd. ⟨http://www.infocraft.lk⟩ is developing a system that uses SHOP2 for material selection in continuous-process manufacturing: specifically, the production of activated carbon from charcoal using a discrete set of continuous manufacturing processes. The desired properties of the carbon (specifically its grade size and adsorption level) will vary from one run to another, as will the characteristics of different supplies of charcoal. The objective of the project is to use SHOP2 to select which supplies of charcoal will most reliably produce activated carbon with a desired set of properties. SHOP2's abilities to do numerical and axiomatic reasoning are essential for this project: adsorption levels are represented as real numbers, and grade sizes are represented as normal distributions.

4.2.4 University Projects

Automated Composition of Web Services (University of Maryland)

Web services are Web accessible, loosely coupled chunks of functionality with an interface described in a machine readable format. Web services are designed to be *composed*, that is, combined in workflows of varying complexity to provide functionality that none of the component services could provide alone.

In the OWL-S (formerly DAML-S) language for semantic markup of web services, services can be described as complex or atomic processes with preconditions and effects. This makes it possible to translate the OWL-S process-model constructs directly to SHOP2 methods and operators, and we have developed an algorithm to do so. This means that SHOP2 can be used to solve service composition problems, by telling SHOP2 to find a plan for the task that is the translation of a composite process [78, 64].

Project Planning (Lehigh University)

The SHOP/CCBR system is a tool developed at the Lehigh University for investigating the use of HTN planning techniques to support project management. The SHOP/CCBR system is a straightforward extension of SHOP that uses *cases* to decompose tasks. Cases are similar in structure to methods, the main difference being that cases include preference information for use in ranking applicable cases. SHOP/CCBR uses a communication module to interact with Microsoft Project, a commercial tool for project management. This allows displaying the HTN decompositions generated with SHOP's hierarchical planning algorithm in Microsoft Project. The on-going work involves developing algorithms to capture cases automatically from user interactions with Microsoft Project.

Statistical Goal Recognition in Agent Systems (University of Rochester)

For an agent to perform effectively in a multi-agent environment, an important task is *goal recognition*, i.e., inferring the goals of other agents. Researchers at the University of Rochester are developing a statistical approach to goal recognition using machine-learning techniques. To do the learning requires a labeled "plan corpus" of plans and their associated goals. They are using SHOP2 to generate such plan corpora stochastically. For this purpose, they are using a modified version of SHOP2 that makes random choices at every point where more than one possible decision is available to SHOP2. For more information about the project, see ⟨http://www.cs.rochester.edu/research/cisd/projects/goalrec⟩.

Additional University Projects

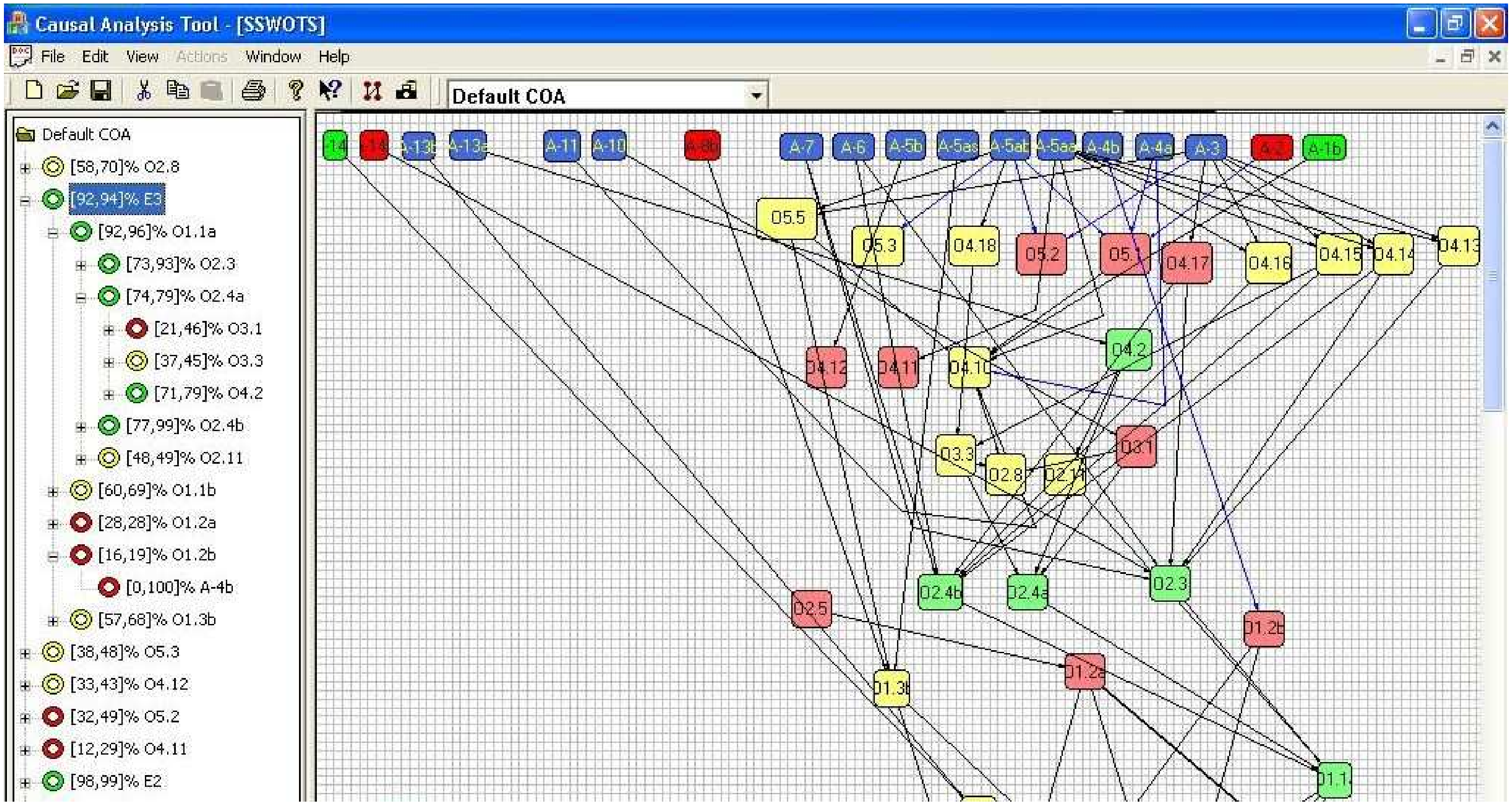
Worldwide, SHOP and SHOP2 have been used in many more college and university projects than we can mention, but here are some notes about a few of them.

- Drexel University regularly uses SHOP and SHOP2 in their Introductory AI class in order to teach planning, and in their Knowledge-Based Agents course to do agent reasoning, service composition, etc.
- At the National University of Colombia in Medellin, Colombia, a system is being developed that uses SHOP2 to automatically create virtual courses from existing educational material.
- At the Technical University of Cluj-Napoca in Romania, SHOP has been used for an e-commerce aplication, to build plans for bidding in a modified version of the Trading Agent Competition.
- At Trinity College Dublin, SHOP2 is being used in a web-service composition project somewhat similar to ours.
- Researchers in the Aerospace Engineering Department at the University of Maryland have just begun a project in which they are using SHOP2 as the planning component in an architecture that combines task planning, real-time scheduling, and motion/trajectory planning.
- At Villanova University, SHOP2 has been used in a mock spacecraft-mission scenario, to study how the density, distribution and overall layout of environment obstacles can be used to compute and predict the best optimization technique to use within SHOP2.

4.3 Results for CAT

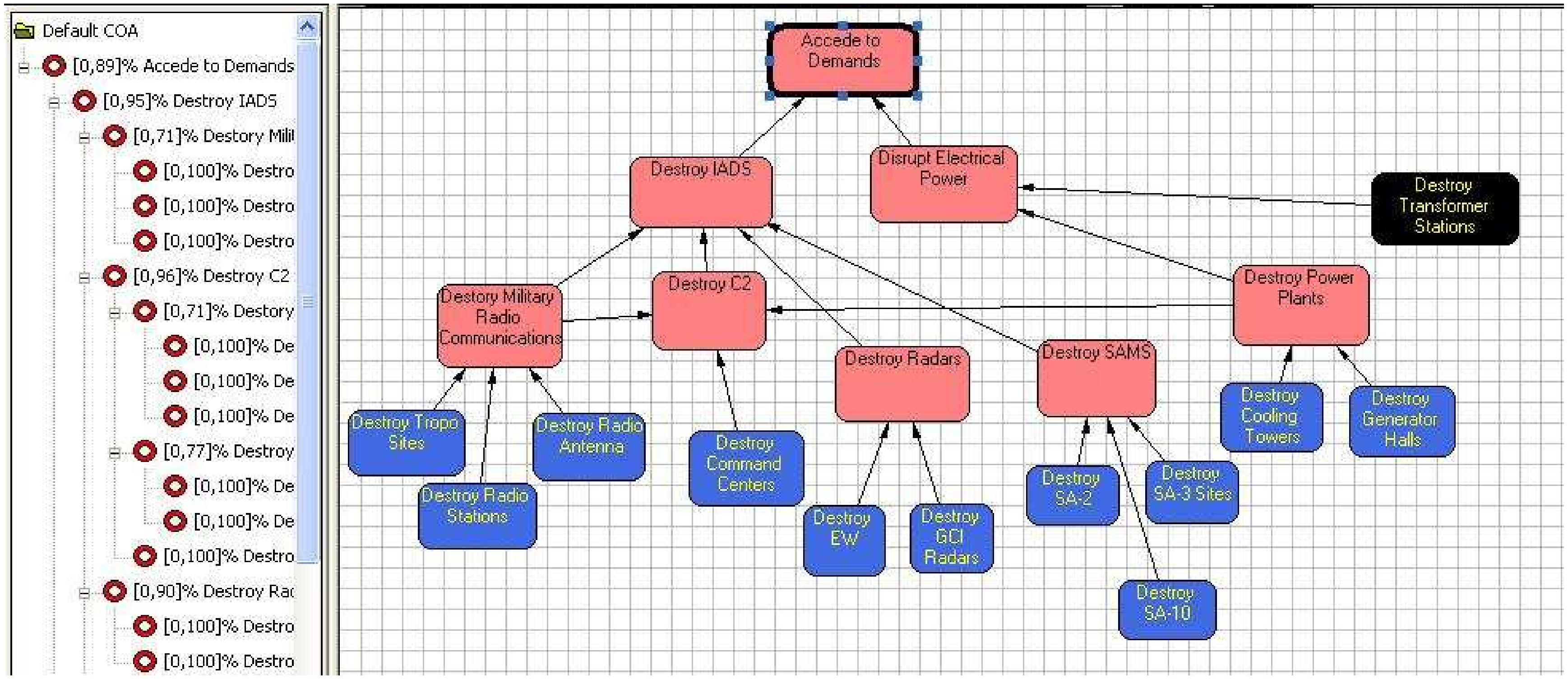
We have implemented our approach in CAT that computes the probabilities and recommendations described in the previous section, and done some preliminary experiments. For our experiments, we have used unclassified versions of causal models for two scenarios. One is the "Operation SSWOTS," a portion of which is shown in Figure 4.4, is a "scrubbed" version of a much larger model developed for the war in Afghanistan. The other, called the "Operation OctMod," shown in Figure 3.7. The "Operation OctMod" model is a representation of the causal model that was used against Milosevic in the Bosnia-Herzegovina war. For each case, it was possible to use our modified version of CAT to develop plans in the order of minutes.

We now describe a sample user session we have performed with the OctMod example. In this example, the maximum and the minimum probabilities of occurrence for the goal event (the "accede to demands" node in Figure 4.5) are 90% and 0%, respectively. The maximum probability of success is achieved when



```
Default COA
  [58,70]% O2.8
  [92,94]% E3
    [92,96]% O1.1a
      [73,93]% O2.3
      [74,79]% O2.4a
        [21,46]% O3.1
        [37,45]% O3.3
        [71,79]% O4.2
      [77,99]% O2.4b
      [48,49]% O2.11
    [60,69]% O1.1b
    [28,28]% O1.2a
    [16,19]% O1.2b
      [0,100]% A-4b
    [57,68]% O1.3b
  [38,48]% O5.3
  [33,43]% O4.12
  [32,49]% O5.2
  [12,29]% O4.11
  [98,99]% E2
```

Figure 4.4: A portion of the causal model for “Operation SSWOTS.”



```
Default COA
  [0,89]% Accede to Demands
    [0,95]% Destroy IADS
      [0,71]% Destory Milit
        [0,100]% Destro
        [0,100]% Destro
        [0,100]% Destro
      [0,96]% Destroy C2
        [0,71]% Destory
          [0,100]% De
          [0,100]% De
          [0,100]% De
        [0,77]% Destroy
          [0,100]% De
          [0,100]% De
        [0,100]% Destro
      [0,90]% Destroy Rac
        [0,100]% Destro
        [0,100]% Destro
```

Figure 4.5: The left-hand pane shows the values computed by our modified version of CAT for the minimum and maximum probabilities of each node in Operation OctMod. Our system recommends performing the rightmost actionable item in the causal network and indicates this by highlighting the node.

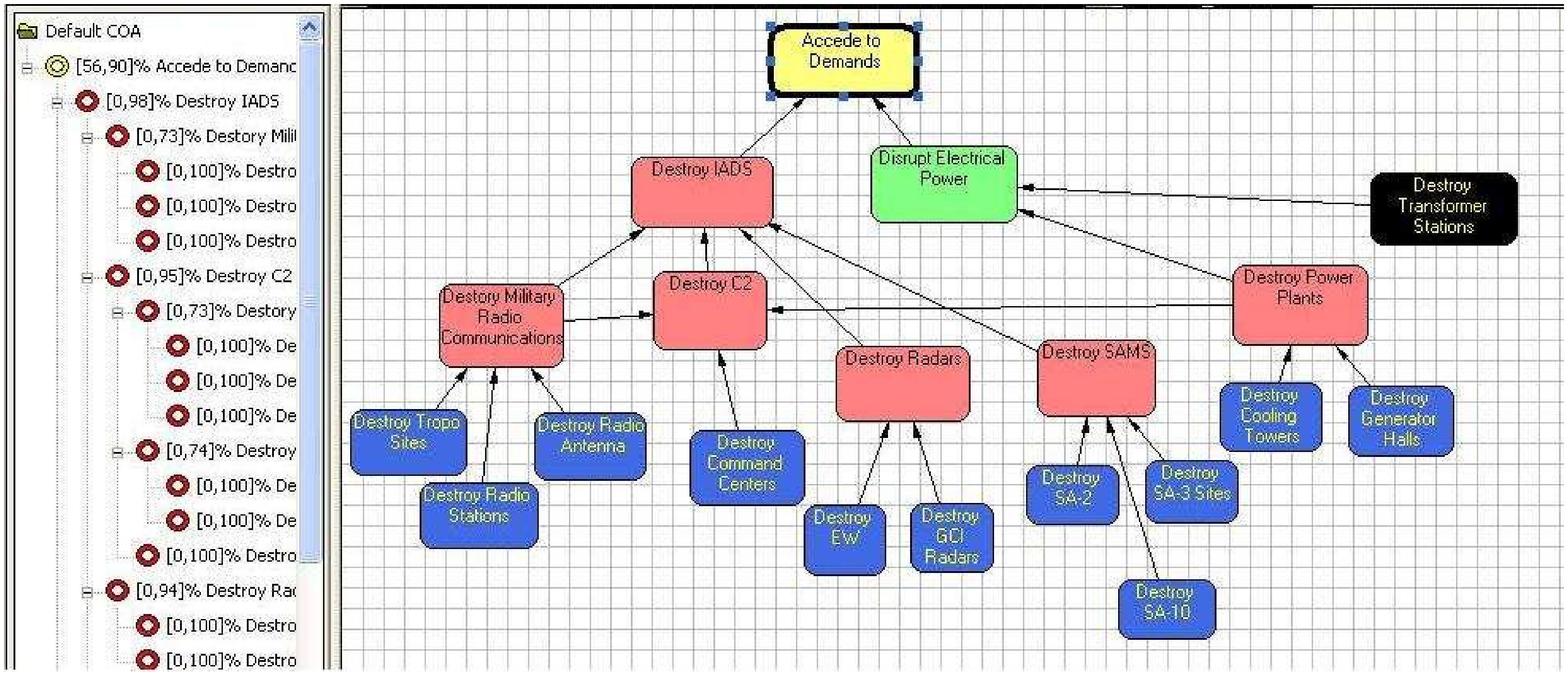


Figure 4.6: If the user decides to follow the recommendation highlighted in Figure 4.5, this substantially increases the minimum probability of achieving the goal.

all of the actions are included in the plan, and the minimum probability of success is achieved when all are excluded.

Initially, we did not specify any decisions on which of the actionable items to include in the plan or exclude from it, so all of the actionable items are marked as *undecided*. We first asked our modified version of CAT to analyze the causal model and make a recommendation. CAT then calculated the maximum and minimum probabilities shown in the left-hand pane in Fig.4.5. Note that these probabilities computed by CAT are correct estimates of the actual minimum and maximum probabilities of the goal node in this example since our approach enables CAT to compute these estimates over every possible combination of the decisions on the undecided actions, virtually in a simultaneous way.

Having computed the estimates of the minimum and maximum probabilities, CAT also calculated that the best choice for us to make next is to include the action "Destroy Transformer Stations," so it highlighted this action in black as shown in Figure 4.5. This action is the one with the greatest estimate of increasing the probability of the goal node. Then we, following CAT's recommendation, included the action in the plan, and asked CAT to analyze the causal model again. As shown in Fig. 4.6, including this action in the plan increases the minimum probability of the goal node from 0% to 56%. At 90%, the node's maximum probability is the same as before except for a 1% difference due to random variation in CAT's simulation. The reason for such an increase in the minimum probability of the goal node is that 56% represents an estimate of the probability of the goal when the recommended action is included in the plan, and the rest of the actions are excluded. The maximum probability of the goal does not change because it is the probability of the goal when all of the actions are included in the plan.

At this point, we again requested a recommendation for what to do next. The iterative planning process continued in this manner until we have made a decision for every actionable item. In the case we followed all of our system's recommendations, the result was a plan whose probability of success is as high as possible (i.e., both minimum and maximum probabilities of the goal is about 90%), in which all of the actionable items are included. The entire process took just a few minutes.

4.4 Related Work

The best general source of information about work on automated planning is [27], which is the first comprehensive textbook on the subject. The following three sections provide a survey of work specifically related to HICAP, SHOP and SHOP2, and CAT.

4.4.1 Work Related to HICAP

Our descriptions of HICAP and SiN are based on [53, 51]. Our additional publications about HICAP include [54, 55, 3, 52, 50, 53]. Below is a summary of other work related to HICAP.

Case-based planners. CHEF [31] and DIAL [42] are case-based, but do not have a generative component, and thus need a large case base to perform well across a wide variety of problems. Prodigy/Analogy [71], DerSNLP [33], and Paris [11] integrate generative and case-based planning, but require a complete domain theory and are not mixed-initiative.

At least three other integrated (case-based/generative), mixed-initiative planners exist. MI-CBP [72], which extends Prodigy/Analogy, limits interaction to providing it with user feedback on completed plans. Thus, it must input, or learn thru feedback, a sufficiently complete domain description to solve problems. In contrast, SiN gathers information it requires from the user through NaCoDAE conversations, but does not learn from user feedback. CAPlan/CbC [56] and Mitchell's [47] system use interaction for plan adaptation rather than to acquire state information.

Among integrated case-based/generative planners, SiN's interleaved control structure is unique in that it allows both subsystems to equally control the task decomposition process. In contrast, other approaches either use heuristics (Prodigy/Analogy, MI-CBP) or order case-based prior to generative planning (DerSNLP, [47]), although Paris does this iteratively through multiple abstraction levels. Distinguishing the relative advantages of these control strategies is an open research issue.

CaseAdvisor [15], like SiN, integrates conversational case retrieval with planning. While CaseAdvisor applies pre-stored hierarchical plans to gather information to solve diagnosis tasks, SiN instead uses its case retriever to gather information and applies cases to refine hierarchical plans.

NEO planning. Although human NEO planners use a number of computer tools as aids to plan development, we know of no deployed NEO planning tools that do any plan generation of their own. [37] proposed a conceptual design for predicting the number and type of personnel required for a NEO. [16] described a decision-theoretic approach for instantiating a general NEO plan with specific parameters for locations, forces, and destinations, and used it to assess alternative plans. [28] presented a system for predicting manning estimates for certain NEO tasks. None of these systems formulate NEO plans. However, [18] proposed a distributed hierarchical planning approach for plan formulation, but this system has not yet been implemented.

Other military planning. Some researchers have reported using case-based planning approaches for HTN planning tasks on military domains. For example, Mitchell [47] used an integrated CBP (case-based planning) approach to select which tasks to be performed for a Tactical Response Planner. NEO planning differs in that requires that *all* tasks must be addressed, and HICAP uses CBP to instead choose *how* to perform a task. MI-CBP [72] uses rationale-directed CBP to suggest plan modifications, but does not perform doctrine-driven task decomposition. HICAP's interactions instead focus on retrieval rather than plan adaptation and learning. IFD4's [12] plan formulation component, ACPT, automatically generates plans as guided by an editable objectives hierarchy. In contrast, HICAP's objectives are fixed, and user interaction focuses on task formulation.

Crisis response planning. Other researchers have described related systems for crisis response tasks. Ferguson and Allen [24] describe an interactive planner for crisis response applications, but the system does not use cases to guide plan formulation. Likewise, the distributed planning tool in [77] does not use cases for plan formulation. [26] describes an interactive hierarchical case-based scheduler for crisis response, but does not perform interactive plan formulation. Avesani *et al.* [6] describe a case-based planning approach for fighting forest fires that supports interactive plan adaptation, but does not use hierarchical guidelines to formulate plans as is done in HICAP. Finally, Leake *et al.* [43] describe a case-based planner applied to disaster response tasks that focuses on learning case adaptation knowledge, but it is not driven by standard requirements and operating procedures, and focuses interaction on knowledge acquisition rather than problem elicitation.

4.4.2 Work Related to SHOP and SHOP2

The basic ideas of HTN planning were first developed more than 25 years ago in work by Sacerdoti [61] and in Tate's Nonlin planner [68].

HTN planning has been more widely used in planning applications than any other automated-planning technique [74]. Examples include production-line scheduling [75], crisis management and logistics [17, 69, 13], planning and scheduling for spacecraft [1, 23], equipment configuration [2], manufacturing process planning [65], evacuation planning [53], the game of bridge [66], and robotics [49, 48].

In a complex application, an HTN planner may generate plans that contain thousands ofnodes. Plans this large are very difficult for humans to understand without a natural pictorial representation. Several HTN planners (e.g., SIPE-2 and O-Plan) provide GUIs to aid in generating plans, viewing them, and following and controlling the planning processes [74]. Particularly useful for visualizing the plan derivation and structure is the ability to view its decomposition tree at various levels of abstraction.

The first steps toward a theoretical model of HTN planning were taken by Yang [79] and Kambhampati [36]. A complete model was developed by Erol [20]. This model provided the basis for the complexity analysis in [22] and the first provably correct HTN planning procedure, UMCP [21].

An alternative model of HTN planning [9, 8] is to use the branching function of classical planning, and consider the methods to be a pruning function. This model is appealing in that it provides a clear relation to classical planning. However, the limitation of this model is that it is only capable of expressing classical planning problems.

The best known domain-independent HTN planning systems are:

- Nonlin [68],[2] one of the first HTN planning systems;
- SIPE-2 [73],[3] which has been used in many application domains;
- O-Plan [17, 69],[4] which has also been used in many application domains;
- UMCP [21],[5] an implementation of the first provably sound and complete HTN planning algorithm;
- SHOP2 [57],[6] which is described in Section 3.2.

High-level effects were first described by Tate [68], and the conditions necessary to achieve soundness with them were explored by Bacchus and Yang [7] and Young *et al.* [80]. The semantics of high-level effects are defined in two different ways in the literature: either as additional effects in addition to the ones asserted by the planning operators (e.g., [7]), or as contraints that must bed satisfied in order for a task to be achieved or a method to succeed (e.g., [20]). These two approaches result in very different planning algorithms.

Declarations of external preconditions have been used in the Nonlin [68] and SIPE-2 [73] planning systems. Algorithms for finding external preconditions automatically have been developed for use in the UMCP system [70].

O-PLAN, SIPE-2, and SHOP2 can each do certain kinds of temporal planning. For details, see the web sites for O-PLAN and SIPE-2 and see [57] for SHOP2.

4.4.3 Work Related to CAT

In this section, we describe some of the knowledge systems that are designed to support military operations, and compare their action-planning techniques with our approach using CAT. We also describe two knowledge-based systems, namely CYPRESS [76] and HICAP (see Section 3.1). that were developed for generating courses of actions under certain conditions of uncertainty.

The CADET system [29] is a knowledge-based tool planning tool that can automatically generate courses of actions. The system is capable of modeling hetergeneous assets and tasks, coordinating team efforts, and generating team action plans in adverserial environments. In that respect, An important difference between our approach with CAT and the CADET system is that, to the best of our knowledge, CADET is not capable of performing probabilistic analyses of cause and effect relationships between the events that may or may not occur during a military operation, and therefore, it is not capable of reasoning about

[2]A copy of Nonlin can be downloaded at http://www.aiai.ed.ac.uk/project/nonlin.

[3]A copy of SIPE-2 can be downloaded at http://www.ai.sri.com/~sipe if the user has a license.

[4]A copy of O-Plan can be downloaded at http://www.aiai.ed.ac.uk/~oplan.

[5]A copy of UMCP can be downloaded at http://www.cs.umd.edu/projects/plus/umcp.

[6]A copy of SHOP2 can be downloaded at http://www.cs.umd.edu/projects/shop.

optimality in generation of the courses of actions.

[60] describes a knowledged-based system for forming coalitions in order to achieve the given objectives. This system, called CPlanT, is an agent-based system in which the agents form alliances according to the information they have about the world and the information they have about the other agents in the world. In this model, the agents prefer to form coalitions within the particular alliances they are involved with, since allied agents know about each other, and therefore, substantial communication overhead is avoided when the coalition is formed within the allience. Once a coalition is formed, team-action planning is done by determining how each team member will contribute to achieving the goals. This task is accomplished by a coordinator agent, which decomposes the goal into subgoals, creates a course of action for each participant agent, and distributes these subgoals to the agents in a contract proposal.

The Coalition Agents Experiment (the CoAX Project) [4] also aims to design an agent-based system for coalition operations. This project aims to provide a rapid integration of agent systems in order to improve interoperability and support human situation awareness without going through a detailed planning process involving the participating agents. Using this system, the human users can develop action plans in various levels of abstraction, and execute those plans in the world. The system also includes agents that represent the other entities in the world other than the coalition members. The behavior of these agents may have an influence on the action plan generated for the coalition members, so the system allows for revising the generated plans, and deconfliction and adjustment of the revised plans via the human users.

A difficulty in our approach might be a practical one: the causal models developed for real military operations could be so huge as to incur too much computational overheadin CAT's probabilistic analysis algorithms. Although this was not the case in our preliminary experiments, the causal models in those examples were rather small. It will be really interesting to test our system with real scenarios and real users, and we intend to do so in the near future.

We are also aware of two knowledge-based systems in which users can perform course-of-action planning in a mixed-initiative way and under certain conditions of uncertainty. CYPRESS [76] is a domain-independent framework for planning in dynamic and uncertain environments. The system is composed of several components responsible for generative planning, reasoning about uncertainty, and plan execution. It is capable of performing both probabilistic and possibilistic (fuzzy-logic based) uncertainty analyses. CYPRESS is similar to our approach in that it uses simulation techniques to compute lower and upper bounds on probabilities. The case representation in HICAP (see Section 3.1 provides a way to reason about certain kinds of uncertainties, but not to reason about probabilities in the way that CAT does.

Chapter 5

Conclusions

The first three sections describe our conclusions regarding HICAP, SHOP and SHOP2, and CAT. The fourth section describes ongoing and future work.

5.1 Conclusions Regarding HICAP

HICAP is an interactive planning tool that assists users in formulating an operational plan. As of the publication date of this report, HICAP had not been deployed, but was still under development at the US Naval Research Laboratory.

HICAP is interactive; it supports task editing and triggers conversations for tasks that can be decomposed in more than one way. The planning process consists of HTN task decomposition, some of which is done by the user using the HTE editor, and some of which is done by HICAP using the SiN planning algorithm. The HTE plan editor allows the user to visually check that all tasks are assigned the necessary resources.

HICAP's SiN procedure is a provably correct procedure for combined case-based and generative planning with incomplete domain descriptions. It tightly integrates NaCoDAE/HTN's case-based task decomposition and JSHOP's generative planning ability. Experimental results with SiN show that a user can dynamically guide SiN by giving preferences to it as part of the user's normal interaction with SiN during the planning process.

SiN's ability to combine both experiential and generative knowledge sources can be beneficial in real-world domains where some processes are well known and others are obscure but recorded memories exists on how they were performed. Evacuation planning is an example of this type of domain.

5.2 Conclusions Regarding SHOP and SHOP2

We have been pleasantly surprised at the extent to which people have begun using SHOP and SHOP2 in their research and development projects. We believe that this has come about for several reasons:

- SHOP and SHOP2 are based on HTN decomposition. The decomposition of tasks into subtasks seems to correspond well to the way in which users think about how to generate plans.
- Unlike most other automated-planning systems, SHOP and SHOP2 plan for tasks in the same order that the tasks will be executed. This removes a great deal of uncertainty at planning time, which makes it easier to write write complex domain descriptions.
- SHOP and SHOP2 are available as open-source software. This has made it easy for users to find and fix bugs, and to adapt the software for their own purposes.

5.3 Conclusions Regarding CAT

We have described a new technique for interactive course-of-action planning under conditions of uncertainty. Our approach is based on the use of CAT (Causal Analysis Tool). CAT was developed by the Air Force Research Laboratory and is in use by a number of military organizations for creating and analyzing causal models.

To do planning in CAT, a user begins with a causal model of the domain in which some of the nodes represent actionable items, and makes decisions about which actions to include in the plan and which not. One of the biggest problems is the exponentially large number of combinations of actionable items: there are far too many of them for users to analyze each one.

To provide a solution to this problem, we have developed a way to quickly compute estimates for the minimum and maximum probabilities of success associated with a partial plan, and use these probabilities to make recommendations about which actions should be included and excluded in order to produce a complete plan with an exponential reduction in the amount of time required. We have implemented this approach in CAT. Our preliminary experiments with this version of CAT showed that our approach looks promising: CAT generated recommendations that produced complete plans with the highest possible probability of success.

We are currently performing an extensive theoretical and experimental analysis of our technique to determine its strengths and its weaknesses. Furthermore, we also intend to extend the technique for reasoning about time. In that respect, we already started extending our implementation in CAT to evaluate our preliminary ideas on probabilistic planning with time. Our ultimate objective is to develop a comprehensive theory of planning with probabilities and time.

5.4 Ongoing and Future Work

The success of the projects described in this report has given us many ideas for improvements and extensions. We now describe some of them.

5.4.1 Automated Learning of Planning Domains

A great challenge in using any planning system to solve real-world problems is the difficulty of acquiring the domain knowledge that the system will need. We have developed ways to address part of this problem by having the planning system *learn* the HTN methods incrementally under supervision of an expert.

We have developed a general formal framework for learning HTN methods, and a supervised learning algorithm, named *CaMeL*, based on this formalism [35]. We have developed theoretical results about CaMeL's soundness, completeness, and convergence properties, and have done experimental studies of its speed of convergence under different conditions. The experimental results suggest that CaMeL may potentially to be useful in real-world applications.

5.4.2 Compiling Planning Domains

A domain-configurable planner may be viewed as an interpreter of its domain-description language: given a domain description D and a planning problem P, the planner invokes the methods and operators of D interpretively on P. An alternative approach is to write a *compiler* for the domain description language: the input to the compiler is a domain description D, and the output is a domain-specific planning program for D, that can be run directly on any planning problem P in D.

The advantages of such a *planner compilation* approach are analogous to the advantages that compilationhas over interpretation in conventional programming languages. By compiling domain descriptions directly into low-level executable code, we can do implementation-level optimizations that are not otherwise possible and have not been explored in previous research on AI planning. These optimizations can be coupled with other speed-up techniques (e.g., domain analysis and other automated domain information synthesis techniques) in order to obtain additional speedups.

We are developing JSHOP2, a Java implementation of SHOP2 that uses this domain-compilation technique. Our preliminary experimental results suggests that the compilation technique substantially increases the planner's efficiency. A technical report on this topic is available [34], and we intend to make JSHOP2 itself available in a few months.

5.4.3 Planning Under Uncertainty

In planning research, the "classical" model of actions is that they have deterministic outcomes. However, in many situations where one might want to do planning, it may be useful to assume that some actions have more than one possible outcome. This action model can be useful in situations where the outcome of an action might vary due to random changes in the environment or to the actions of other agents.

We are developing a general technique for taking forward-chaining planners for deterministic domains and adapting them to work in nondeterministic planning domains, i.e., planning domains in which each action may have more than one possible outcome. We have shown both theoretically and experimentally that our approach can produce exponential speedups over previous algorithms for planning in nondeterministic environments [38].

We are currently extending our approach to work for situations in which actions have probabilistic outcomes (e.g., MDP models of actions). We believe that we will be able to obtain similar speedups in these kinds of planning domains.

5.4.4 Planning with Distributed Information Sources

Planning researchers typically assume that the planning system is *isolated*: it begins with a complete description of the planning problem, and has no need of interacting with the external world during the planning process. In many practical situations, such an assumption is clearly unrealistic: the planner may need to obtain information from external sources during planning.

We have developed a formalism for *wrappers* that may be placed around conventional (isolated) planners to replace some of the planner's memory accesses with queries to external information sources. When appropriate, the wrapper can automatically backtrack the planner to a previous point in its operation. We have done both mathematical and experimental analysis of several different query-management strategies for these wrappers, i.e., strategies for when to issue queries, and when/how to backtrack the planner. Our results [5] show conditions under which different query management strategies are preferable, and suggest that domain-configurable planners such as SHOP2 are likely to be better suited than other planners for planning with volatile information.

Even better performance can be obtained if a planner can make *non-blocking* queries to external information sources, i.e., if the planner can continue exploring other parts of its search space while waiting for the response to a query. We have developed a modified version of SHOP2 that works in this way. We have shown experimentally that this dramatically improves (i) the time needed to find a solution and (ii) in cases where the information source is not guaranteed to respond, the chance of finding a solution at all [39, 40].

List of Acronyms

AHEAD:	Analogical Hypothesis Elaborator for Activity Detection http://www.nrl.navy.mil/aic/iss/ida/projects/ahead/AHEAD.php
CADET:	Course of Action Development and Evaluation Tool http://cadet.bbn.com/
CaMeL:	Candidate Elimination Method Learner http://www.cs.umd.edu/~nau/papers/camel-aips2002.pdf
CYPRESS:	the name of an integrated planning system; see http://www.ai.sri.com/~cypress/tucson/tucson.html
HTE:	Hierarchical Task Editor http://www.aic.nrl.navy.mil/hicap/
HICAP:	Hierarchical Interactive Case-based Architecture for Planning http://www.aic.nrl.navy.mil/hicap/
IADS:	Integrated Air Defense Systems
MDP:	Markov Decision Process
NaCoDAE:	Navy Conversational Decision Aids Environment http://home.earthlink.net/~dwaha/software/nacodae/
SIFT, LLC:	Smart Information Flow Technologies, LLC http://www.sift.info/
SHOP:	Simple Hierarchical Ordered Planner http://www.cs.umd.edu/projects/shop
SHOP2:	Simple Hierarchical Ordered Planner 2 http://www.cs.umd.edu/projects/shop
SiN:	SHOP in NaCoDAE http://www.aic.nrl.navy.mil/hicap/pubs-pa.html

Bibliography

[1] M. Aarup, M. M. Arentoft, Y. Parrod, J. Stader, and I. Stokes. OPTIMUM-AIV: A knowledge-based planning and scheduling system for spacecraft AIV. In *Intelligent Scheduling*, pages 451–469. Morgan Kaufmann, 1994.

[2] J. M. Agosta. Formulation and implementation of an equipment configuration problem with the SIPE-2 generative planner. In *Proc. AAAI-95 Spring Symposium on Integrated Planning Applications*, pages 1–10, 1995.

[3] D. W. Aha, L. A. Breslow, H. Muñoz-Avila, D. S. Nau, and R. Weber. HICAP: Hierarchical interactive case-based architecture for planning, 2000.

[4] D. Allsopp, P. Beautement, J. M. Bradshaw, E. H. Durfee, M. Kirton, C. A. Knoblock, N. Suri, A. Tate, and C. W. Thompson. Coalition agents experiment: Multiagent cooperation in international coalitions. *IEEE Intelligent Systems*, 17(3):26–35, 2002.

[5] T.-C. Au, D. Nau, and V. Subrahmanian. Utilizing volatile external information during planning. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 647–651, August 2004.

[6] P. Avesani, A. Perini, and F. Ricci. The twofold integration of CBR in decision support systems. In D. Aha and J. Daniels, editors, *Case-Based Reasoning Integrations: Papers from the 1998 Workshop*, number WS-98-15. AAAI Press, 1998.

[7] F. Bacchus and Q. Yang. The downward refinement property. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 286–292, 1991.

[8] A. Barrett. *Frugal Hierarchical Task-Network Planning*. PhD thesis, University of Washington, 1997.

[9] A. Barrett and D. S. Weld. Task-decomposition via plan parsing. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, volume 2, pages 1117–1122, Seattle, Washington, USA, 1994. AAAI Press/MIT Press.

[10] M. Beetz, J. Hertzberg, M. Ghallab, and M. Pollack, editors. *Advances in plan-based control of robotics agents*, volume 2466. LNAI, Springer-Verlag, 2002.

[11] R. Bergmann and W. Wilke. Building and refining abstract planning cases by change of representation language. *Journal of Artificial Intelligence Research*, 3:53–118, 1995.

[12] M. Bienkowski and L. Hoebel. Integrating ai components for a military planning application. In *AAAI/IAAI Proceedings*, pages 561–566. AAAI Press, 1998.

[13] S. Biundo and B. Schattenberg. From abstract crisis to concrete relief – a preliminary report on combining state abstraction and HTN planning. In *Proceedings of the European Conference on Planning (ECP)*, pages 157–168, 2001.

[14] L. A. Breslow and D. W. Aha. NACODAE: Navy conversational decision aids environment. Technical Report AIC-97-018, Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, 1997.

[15] C. Carrick, Q. Yang, I. Abi-Zeid, and L. Lamontagne. Activating CBR systems through autonomous information gathering. In *Proceedings of the Third International Conference on Case-Based Reasoning*, pages 74–88. Springer, 1999.

[16] T. Chavez and M. Henrion. Focusing on what matters in plan evaluation: Efficiently estimating the value of information. In *Proceedings of the ARPA/Rome Laboratory Knowledge-Based Planing and Scheduling Initiative*, pages 387–399. Morgan Kaufmann, 1994.

[17] K. Currie and A. Tate. O-Plan: The open planning architecture. *Artificial Intelligence*, 52(1):49–86, 1991.

[18] M. desJardins, A. Francis, and M. Wolverton. Hybrid planning: An approach to integrating generative and case-based planning. In D. Aha and J. Daniels, editors, *Case-Based Reasoning Integrations: Papers from the 1998 Workshop*, number WS-98-15. AAAI Press, 1998.

[19] DoD. Joint tactics, techniques and procedures for noncombat evacuation operations. Technical Report Joint Publication Report 3-07.51 Second Draft, 1994.

[20] K. Erol, J. Hendler, and D. S. Nau. Semantics for hierarchical task-network planning. Technical Report CS TR-3239, UMIACS TR-94-31, ISR-TR-95-9, University of Maryland, March 1994.

[21] K. Erol, J. Hendler, and D. S. Nau. UMCP: A sound and complete procedure for hierarchical task-network planning. In *Proceedings of the International Conference on AI Planning Systems (AIPS)*, pages 249–254, June 1994.

[22] K. Erol, J. Hendler, and D. S. Nau. Complexity results for hierarchical task-network planning. *Annals of Mathematics and Artificial Intelligence*, 18:69–93, 1996.

[23] T. A. Estlin, S. Chien, and X. Wang. An argument for a hybrid HTN/operator-based approach to planning. In *Proceedings of the European Conference on Planning (ECP)*, pages 184–196, 1997.

[24] G. Ferguson and J. Allen. TRIPS: An integrated intelligent problem-solving assistant. In *AAAI/IAAI Proceedings*, pages 567–572, 1998.

[25] M. Fox and D. Long. International planning competition, 2002. http://www.dur.ac.uk/d.p.long/competition.html.

[26] M. Gervasio, W. Iba, and P. Langley. Case-based seeding for an interactive crisis response assistant. In D. Aha and J. Daniels, editors, *Case-Based Reasoning Integrations: Papers from the 1998 Workshop*, number WS-98-15. AAAI Press, 1998.

[27] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, May 2004.

[28] Y. Gil, M. Hoffman, and A. Tate. Domain-specific criteria to direct and evaluate planning systems. In *Proceedings of the ARPA/Rome Laboratory Knowledge-Based Planing and Scheduling Initiative*, pages 433–444, Tuscon, AR, 1994. Morgan Kaufmann.

[29] L. Ground, A. Kott, and R. Budd. A knowledge-based tool for planning of military operations: The coalition perspective. In *Proceedings of the Second International Conference on Knowledge Systems for Coalition Operations*, pages 195–203, Toulouse, France, 2002.

[30] K. J. Hammond. Learning to anticipate and avoid planning problems through the explanation of failures. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1986.

[31] K. J. Hammond. *Case-Based Planning: viewing learning as a memory task*. Academic Press, New York, 1989.

[32] M. Henrion. Propagating uncertainty by logic sampling in bayesian networks. Technical report, Department of Engineering and Public Policy, Carnegie-Mellon University, 1986.

[33] L. Ihrig and S. Kambhampati. Derivational replay for partial order planning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 992–997. AAAI Press, 1994.

[34] O. Ilghami and D. S. Nau. A general approach to synthesize problem-specific planners. Technical Report CS-TR-4597, UMIACS-TR-2004-40, University of Maryland, October 2003.

[35] O. Ilghami, D. S. Nau, H. Muñoz-Avila, and D. W. Aha. CaMeL: Learning methods for HTN planning. In *AIPS-2002*, Toulouse, France, 2002.

[36] S. Kambhampati and J. H. A. A validation structure based theory of plan modification and reuse. *Artificial Intelligence*, 55:193–258, 1992.

[37] S. Kostek. A user's design of a decision support system for noncombatant evacuation operations for united states central command. Master's thesis, 1988.

[38] U. Kuter and D. Nau. Forward-chaining planning in nondeterministic domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 513–518, July 2004.

[39] U. Kuter, E. Sirin, D. Nau, B. Parsia, and J. Hendler. Information gathering during planning for web services composition. In *ICAPS-04 Workshop on Planning and Scheduling for Web and Grid Services*, 2004.

[40] U. Kuter, E. Sirin, D. Nau, B. Parsia, and J. Hendler. Information gathering during planning for web services composition. In F. v. H. Sheila A. McIlraith, Dimitris Plexousakis, editor, *3rd International Semantic Web Conference (ISWC2004)*, volume 3298, pages 335–349. Springer-Verlag, 2004.

[41] K. S. Lambert. Noncombatant evacuation operations: Plan now or pay later. Technical report, Naval War College, 1992.

[42] D. Leake, A. Kinley, and D. Wilson. A case study of case-based cbr. In *Proceedings of the International Conference on Case-Based Reasoning (ICCBR)*, pages 371–382. Springer, 1997.

[43] D. B. Leake, A. Kinley, and D. Wilson. Acquiring case adaptation knowledge: A hybrid approach. In *AAAI/IAAI Proceedings*, pages 684–689. AAAI Press, 1996.

[44] J. F. Lemmer. Causal modeling. In *Ninth International Conference on Uncertainty In AI (UAI-93)*. Morgan Kaufmann, 1993.

[45] J. F. Lemmer. The causal markov condition: Fact or artifact? *SIGART*, 7(3), 1996.

[46] J. F. Lemmer and D. Gossink. Recursive noisy-or: A rule for estimating complex probabilistic causal interactions. *IEEE Transactions on Systems, Man, and Cybernatics*, 2004. Reference Number: SMCB-E-10152003-0493.R1, To appear.

[47] S. Mitchell. A hybrid architecture for real-time mixed-initiative planning and control. In *AAAI/IAAI Proceedings*, pages 1032–1037, 1997.

[48] B. Morisset and M. Ghallab. *Learning how to combine sensory-motor modalities for a robust behavior*, pages 157–178. Volume 2466 of Beetz et al. [10], 2002.

[49] B. Morisset and M. Ghallab. Synthesis of supervision policies for robust sensory-motor behaviors. In *7th International Conference on Intelligent Autonomous Systems*, pages 236–243, 2002.

[50] H. Muñoz-Avila. Case-base maintenance by integrating case index revision and case retention policies in a derivational replay framework. *Computational Intelligence*, 17(2), 2001.

[51] H. Muñoz-Avila, D. W. Aha, L. Breslow, and D. S. Nau. HICAP: an interactive case-based planning architecture and its application to noncombatant evacuation operations. In *AAAI/IAAI Proceedings*, pages 870–875, 1999.

[52] H. Muñoz-Avila, D. W. Aha, L. A. Breslow, D. S. Nau, and R. Weber. Integrating conversational case retrieval with generative planning. In *EWCBR-2000*, Trento, Italy, 2000. Springer-Verlag.

[53] H. Muñoz-Avila, D. W. Aha, D. S. Nau, R. Weber, L. Breslow, and F. Yaman. SiN: Integrating case-based reasoning with task decomposition. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Seattle, August 2001.

[54] H. Muñoz-Avila, L. Breslow, D. W. Aha, and D. S. Nau. Description and functionality of neodocta. Technical Report AIC-96-005, Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, 1998.

[55] H. Muñoz-Avila, D. McFarlane, D. W. Aha, J. Ballas, L. Breslow, and D. S. Nau. Using guidelines to constrain interactive case-based HTN planning. In *Proceedings of the International Conference on Case-Based Reasoning (ICCBR)*, pages 288–302, 1999. Finalist for the best paper award.

[56] H. Muñoz-Avila and F. Weberskirch. Planning for manufacturing workpieces by storing, indexing and replaying planning decisions. *Proceedings of the International Conference on AI Planning Systems (AIPS)*, 1996.

[57] D. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, 20:379–404, December 2003.

[58] D. S. Nau, Y. Cao, A. Lotem, and H. Muñoz-Avila. SHOP: Simple hierarchical ordered planner. In T. Dean, editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 968–973. Morgan Kaufmann Publishers, July 31–August 6 1999.

[59] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Fransisco, CA, 1988.

[60] M. Pechoucek, V. Marik, and J. Barta. Knowledge based approach to ootw coalition formation. *IEEE Intelligent Systems*, 17(3):17–25, 2002.

[61] E. Sacerdoti. The nonlinear nature of plans. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 206–214, 1975. Reprinted in [**?**], pp. 162–170.

[62] A. B. Siegel. Eastern exit: The noncombatant evacuation operation (neo) from mogadishu, somalia, in january 1991. Technical Report CRM 91-221, Center for Naval Analyses, 1991.

[63] A. B. Siegel. Requirements for humanitarian assistance and peace operations: Insights from seven case studies. Technical Report CRM 94-74, Center for Naval Analyses, 1995.

[64] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau. HTN planning for web service composition using SHOP2. *Journal of Web Semantics*, 2004. To appear.

[65] S. J. J. Smith, D. S. Nau, and T. Throop. Bridge baron, the 1997 world champion of computer bridge, uses AI planning techniques to plan its declarer play, 1997.

[66] S. J. J. Smith, D. S. Nau, and T. Throop. Computer bridge: A big win for AI planning. *AI Magazine*, 19(2):93–105, 1998.

[67] D. T. Stahl. Noncombatant evacuation operations in support of the national military strategy. Technical report, US Army Command and General Staff College, School of Advanced Military Studies, 1992.

[68] A. Tate. Generating project networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 888–893, 1977.

[69] A. Tate, B. Drabble, and R. Kirby. *O-Plan2: An Architecture for Command, Planning and Control*. Morgan-Kaufmann, 1994.

[70] R. Tsuneto, J. Hendler, and D. S. Nau. Analyzing external conditions to improve the efficiency of HTN planning. In *AAAI/IAAI Proceedings*, pages 913–920, 1998.

[71] M. M. Veloso. *Planning and learning by analogical reasoning*. Springer-Verlag, 1994.

[72] M. M. Veloso, A. M. Mulvehill, and M. Cox. Rationale-supported mixed-initiative case-based planning. In *AAAI/IAAI Proceedings*, pages 1072–1077. AAAI Press, 1997.

[73] D. Wilkins. Can AI planners solve practical problems? *Computational Intelligence*, 6(4):232–246, 1990.

[74] D. Wilkins and M. desJardins. A call for knowledge-based planning. *AI Magazine*, 22(1):99–115, 2001.

[75] D. E. Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm*. Morgan Kaufmann, San Mateo, CA, 1988.

[76] D. E. Wilkins. Planning and reacting in uncertain and dynamic environments. *Journal of Experimental and Theoretical AI*, 7(1):197–227, 1995.

[77] M. Wolverton and M. desJardins. Controlling communication in distributed planning using irrelevance reasoning. In *AAAI/IAAI Proceedings*, pages 868–874, 1998.

[78] D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating DAML-S web services composition using SHOP2. In *Proceedings of the Second International Semantic Web Conference (ISWC2003)*, November 2003.

[79] Q. Yang. Formalizing planning knowledge for hierarchical planning. *Computational Intelligence*, 6(1):12–24, 1990.

[80] R. M. Young, M. E. Pollack, , and J. D. Moore. Decomposition and causality in partial-order planning. In *Proceedings of the International Conference on AI Planning Systems (AIPS)*, 1994.